



Code d'éléments finis pour la résolution de problèmes d'équations aux dérivées partielles

Dominique Sandri

► To cite this version:

Dominique Sandri. Code d'éléments finis pour la résolution de problèmes d'équations aux dérivées partielles. 2011. [hal-04124784](#)

HAL Id: hal-04124784

<https://hal.science/hal-04124784v1>

Preprint submitted on 16 Jun 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY-NC-ND 4.0 - Attribution - Non-commercial use - No
Derivative Works - International License



PREPUBLICATION

de
l'Institut Camille Jordan

U.M.R. 5208 du CNRS
Mathématiques et applications

Code d'éléments finis pour la résolution de problèmes d'équations aux dérivées partielles

Seconde partie : algorithmes et description technique du code.

Appendice.

Dominique Sandri

Janvier 2011
N° 320

**Université de Lyon
Université Lyon 1
Ecole Centrale de Lyon
INSA de Lyon
CNRS UMR 5208
Institut Camille Jordan**

Code d'éléments finis pour la résolution de problèmes d'équations aux dérivées partielles

D. Sandri, Université Lyon 1

Institut Camille Jordan, Lyon - Laboratoire Jean Kuntzmann, Grenoble (2007)

Seconde partie : algorithmes et description technique du code.

Appendice.

4. Description du code

Au Paragraphe 4.1, nous donnons la description générale des fonctions du code et de l'algorithme du code.

Nous avons détaillé les fonctions d'assemblage :

- EF_assemblage,
- EF_Prepa,
- EF_Prepa_don,

ainsi que la fonction

- path_EF

du répertoire EF ainsi que les fonctions des sous-répertoires de EF :

- Assemblage,
- Assemblage_elementaire
- deg_lib,
- mat_cell,
- trace_graphe_interp
- trait_cb.

Pout l'utilisation des fonctions du répertoire de maillage

- trait_maillage

et du répertoire

- utilitaires,

non traitées dans ce Paragraphe, on consultera les Paragraphes 2.4, 5.17 et 2.5 pour leur utilisation.

Nous n'avons pas détaillé le cas des EF adaptatifs dont l'utilisation est décrite au Paragraphe 2.11.8 et dont les fonctions placées dans le répertoire

- EF/adapt

permettent principalement des calculs de norme qui prennent en compte en particulier les dérivées par morceaux d'ordre 2 de fonctions de l'espace EF produit E ainsi que les sauts de dérivées à travers les arêtes des triangles.

Nous n'avons pas détaillé le cas des EF courbes dont le principe est analogue au cas des EF non courbes. Le traitement des EF courbe fait l'objet d'un répertoire

- EF/element_t

spécifique qui tient compte des transformations non affines des triangles et qui reprend les fonctions pour des répertoires ci-dessus. L'utilisation des EF courbes, qui prennent aussi en compte l'adaptivité, est décrite au Paragraphe 2.12 précédent.

Le Paragraphe 4.2 est une description des variables principales utilisées dans le code et qui reviennent fréquemment ; variables de maillage, de système, solutions,

4.1.1. LISTE DES REPERTOIRES ET FONCTIONS

etc. Le Paragraphe 4.3 est une description des fonctions principales du code : on donne leur algorithme, le cas échéant le principe mathématique sous-jacent et des exemples d'utilisation : exemple tracé des solutions, formules de quadrature, assemblage, etc. Pour le cas adaptatif et le cas EF courbes on consultera directement le **help** des fonctions et les exemples vus aux Paragraphes précédents.

4.1. Fonctions et algorithme général du code.

Dans ce Paragraphe on donne une liste des répertoires et fonctions utilisées par le code ainsi que l'algorithme général du code. Dans les noms de fonctions, les termes `_T_`, `_A_` et `_Ad_` signifient que les calculs sont relatifs respectivement aux triangles, aux arêtes frontières et aux arêtes.

4.1.1. Liste des répertoires et fonctions utilisés par le code.

On donne ci-dessous les fonctions qui composent le code. On donne les fonctions parentes quand celles-ci concernent la partie assemblage du code EF ou les fonctions de tracé. Nous n'indiquons pas les fonctions parentes des fonctions de calculs de normes qui sont des fonctions utilisées directement par l'utilisateur, et aussi, le code pouvant être traité de manière indépendamment des deux répertoires **adapt** (adaptatif) et **element_t** (EF courbes) si on ne fait pas de calculs en adaptatif ou sur des EF courbes, des fonctions de ces répertoires. On note une exception à cette indépendance qui est la fonction utilisateur **valeur_fonc_quad_K** décrite ci-dessous qui utilise des fonctions du répertoire **adapt**.

Fonctions principales de pré-assemblage et d'assemblage

Nom	Répertoire	Parent
EF_Prepa	EF	
EF_assemblage	EF	
EF_prepa_don	EF	EF_Prepa
path_EF	EF	Fonction indépendante du code
path_EF_unix	EF	Fonction de path utilisateur adaptée a l'environnement

La fonction **path_EF** est la seule fonction à modifier en fonction de l'environnement informatique utilisé en indiquant dans cette fonction l'emplacement des répertoires du code. On pourra aussi créer une fonction qui la remplace comme la fonction **path_EF_unix** donnée en exemple.

Fonctions d'assemblage élémentaire

Nom	Sous Répertoire de EF	Parent(s)
<i>Calcul du fichier relatif aux espaces d'EF et aux formules de quadrature :</i>		
matrice_elem_prepa_T	Assemblage_elementaire	EF_prepa_don
matrice_elem_prepa_A	Assemblage_elementaire	(")

matrice_elem_prepa_Ad Assemblage_elementaire (")

Calcul des matrices élémentaires pour l'assemblage du second membre :

matrice_elem_T_scm Assemblage_elementaire EF_assemblage
 matrice_elem_A_scm Assemblage_elementaire (")
 matrice_elem_Ad_scm Assemblage_elementaire (") en développement,
 fonctionne

Calcul des matrices élémentaires pour l'assemblage de la matrice du système :

matrice_elem_T Assemblage_elementaire EF_assemblage
 matrice_elem_A Assemblage_elementaire (")
 matrice_elem_Ad Assemblage_elementaire (") en développement,
 fonctionne

Cas de la méthode d'EF discontinus pour le transport :

matrice_elem_Ad_int_LR Assemblage_elementaire EF_assemblage
 EF_assemblage_t

*Calculs préalables au calcul des matrices/seconde membres élémentaires
 d'assemblage :*

pre_assemble_T Assemblage_elementaire matrice_elem_prepa_T
 matrice_elem_T
 matrice_elem_T_scm
 pre_assemble_A Assemblage_elementaire matrice_elem_prepa_A
 matrice_elem_prepa_Ad
 matrice_elem_A
 matrice_elem_Ad
 matrice_elem_Ad_int_LR
 matrice_elem_A_scm
 matrice_elem_Ad_scm

*Coefficients et coordonnées cartésiennes et barycentriques des points des formules
 de quadrature :*

assemble_coeff_T Assemblage_elementaire matrice_elem_prepa_T
 matrice_elem_T
 matrice_elem_T_scm
 matrice_elem_prepa_T_t
 matrice_elem_T_t
 matrice_elem_T_scm_t
 assemble_coeff_A Assemblage_elementaire matrice_elem_prepa_A
 matrice_elem_prepa_Ad
 matrice_elem_A
 matrice_elem_Ad
 matrice_elem_Ad_int_LR

4.1.1. LISTE DES REPERTOIRES ET FONCTIONS

matrice_elem_A_scm
matrice_elem_Ad_scm

Fonctions de base :

F_base_T	Assemblage_elementaire	pre_assemble_T valeur_fonc_EF trac_solEc2, trac_solEd2 interp_EF2, interp_EF_p2 interp_EF_ddl2, inv_interp_EF_ddl2
F_base_A	Assemblage_elementaire	pre_assemble_A

Dans la version actuelle du code les fonctions F_base_T et F_base_A sont identiques, elles diffèrent dans leur version élément finis courbes du répertoire element.t. Ces deux fonctions ont en commun la fonction F_base suivante qui effectue en fonction de i et j le calcul effectif de la valeur aux points de la formule de quadrature utilisée des fonctions de base de l'espace $P_{i,j}(K)$ défini en (1.20) au Paragraphe 1.4. Il est à noter que la fonction F_base est une des fonctions principales du code.

F_base	Assemblage_elementaire	F_base_T F_base_A trac_solEc, trac_solEd interp_EF, interp_EF_p interp_EF_ddl, inv_interp_EF_ddl

La fonction F_base_der2 ci-dessous est la version de F_base qui prend en compte les dérivées d'ordre 2 utilisées notamment pour le cas adaptatif. Cette fonction n'est pas utilisée directement dans le code : pour l'adaptatif c'est la fonction F_base_adapt qui est utilisée et pour les EF courbes adaptatif c'est la fonction F_base_adapt.t. Dans la version actuelle du code ces trois fonctions sont identiques. La fonction F_base_der2_3 qui suit est une version de F_base_der2 où les espaces $P_{i,j}(K)$ sont de la forme la forme $P_{i,j,k}(K)$, où i est le degré des fonctions de $P_i(K)$ dont on garde les fonctions attachées aux *ddl* placés sur les sommets, où j est le degré des fonctions de $P_j(K)$ dont on garde les fonctions attachées aux *ddl* placés sur les arêtes, hors sommets, et où k est le degré des fonctions de $P_k(K)$ dont on garde les fonctions attachées aux *ddl* placés à l'intérieur du triangle K .

F_base_der2	Assemblage_elementaire	(non utilisée)
F_base_der2_3	Assemblage_elementaire	(non utilisée)

Traitement de l'assemblage élémentaire du système :

assemble_equ	Assemblage_elementaire	matrice_elem_T matrice_elem_A matrice_elem_Ad matrice_elem_T_scm

		matrice_elem_A_scm
		matrice_elem_Ad_scm
equationJ	Assemblage_elementaire	assemble_equ

La fonction utilisateur `valeur_fonc_quad_K` donne la valeur d'une fonction aux points de la formule de quadrature d'un triangle. Cette fonction utilise la fonction `pre_assemble_T_adapt` du répertoire `adapt` au lieu de la fonction `pre_assemble_T`, ceci afin de permettre la prise en compte des dérivées jusqu'à l'ordre 2 des fonctions :

Fonction utilisateur

<code>valeur_fonc_quad_K</code>	<code>Assemblage_elementaire</code>	fonction utilisateur

Sous-fonctions

<code>mult_cp_prep</code>	Sous fonction de	<code>assemble_equ</code>
<code>mult_cp</code>	Sous fonction de	<code>equationJ</code>

Utilitaires de tracé, d'interpolation et de projection, tracés

Nom	Répertoire	Parent
<i>Calcul des degrés de liberté :</i>		
<code>ddl</code>	<code>trace_graphe_interp</code>	

Valeur aux ddl des fonctions d'après leurs composantes et réciproque :

<code>interp_EF_ddl</code>	(")
<code>inv_interp_EF_ddl</code>	(")

Fonctions identiques aux précédentes mais utilisent `F_base_T` au lieu de `F_base` :

<code>interp_EF_ddl2</code>	(")
<code>inv_interp_EF_ddl2</code>	(")

Interpolation polynômiale des fonctions d'après leurs composantes :

<code>interp_EF</code>	(")
<code>interp_EF_p</code>	(")

Fonctions identiques aux précédentes mais utilisent `F_base_T` au lieu de `F_base` :

<code>interp_EF2</code>	(")
<code>interp_EF_p2</code>	(") (<i>Comme la précédente mais ne trace pas</i>)

Projection de fonctions et des dérivées, tracés :

<code>proj_deriv_EF</code>	(")
<code>proj_deriv_EF2</code>	(") (<i>interpole et projette sur un deuxième espace</i>)
<code>proj_deriv_EF3</code>	(") (<i>projette sur un deuxième espace</i>)
<code>proj_deriv_EF2b</code>	(") (<i>même syntaxe que <code>proj_deriv_EF3</code> et mêmes calculs que <code>proj_deriv_EF2</code></i>)

4.1.1. LISTE DES REPERTOIRES ET FONCTIONS

Tracé de fonctions avec et sans pdetool :

trac_solEc (")
trac_solEd (")

Fonctions identiques aux précédentes mais utilisent F_base_T au lieu de F_base :

trac_solEc2 (")
trac_solEd2 (")

Coupes, utilisé aussi pour les prolongements, utilise la fonction pt_tri :

trac_sol_coup (") proj_deriv_EF2

Fonctions diverses :

interp_2D	(")	interp_EF, interp_EF2 interp_EF_p, interp_EF_p2
maillage_KR	(")	trac_solEc, trac_solEd trac_solEc2, trac_solEd2 interp_EF, interp_EF2 interp_EF_p, interp_EF_p2
coord_maillage_KB	(")	(")

maillage_KR est appelée dans les exemples de coord_maillage_KB, interp_2D

coord_maillage_KB est appelée dans les exemples de maillage_KR, interp_2D

La fonction `maillage_KR` permet de subdiviser un triangle de référence en M^2 sous-triangles. La fonction `coord_maillage_KB` permet de passer des coordonnées barycentriques aux coordonnées cartésiennes. La fonction `interp_2D` calcule des polynômes d'interpolation à deux variables.

Valeur d'une fonction en des points d'un triangle donné à partir de ses composantes dans la base canonique :

valeur_fonc_EF (") trac_sol_coup

Correspondance entre des points quelconques et leur triangle d'appartenance :

pt_tri (") trac_sol_coup

Fonctions diverses hors du répertoire appelées par la fonction ddl :

<code>maillage_ddl</code>	<code>deg_lib</code>	<code>ddl</code>
<code>maillage_triangle_arete_ps</code>	<code>deg_lib</code>	<code>ddl</code>

Utilitaire de vérification des données

Nom	Sous rép. de EF Parent(s)

verif_don_EF	utilitaires	fonction utilisateur
verif_don_EF_warn	utilitaires	(")

Calcul des degrés de liberté

Nom	Sous Rép. de EF	Parent(s)
maillage_ddl	deg_lib	EF_prepa_don, ddl
maillage_triangle_arete_ps	deg_lib	EF_prepa_don, ddl
maillage_triangle_arete	deg_lib	maillage_triangle_arete_ps
triangle_arete	deg_lib	maillage_triangle_arete

Calcul sur les matrices assemblées

Nom	Répertoire	Parent(s)
<i>Traitement des conditions au bord, changements de base :</i>		
cb	EF/trait_cb	EF_Asemblage
systeme_bord_cell	EF/Assemblage	EF_Asemblage
changt_base_nt	EF/Assemblage	EF_Asemblage
changt_base_a	EF/Assemblage	EF_Asemblage
base_nt	EF/Assemblage	changt_base_nt
front_ddl	EF/Assemblage	changt_base_nt
transf_mat1	EF/Assemblage	systeme_bord_cell

Calcul de normes ou d'intégrales (pour int_fonc_T) :

norm_energ_cell	EF/Assemblage	fonction utilisateur
form_bilin_cell	EF/Assemblage	fonction utilisateur
norm_A	EF/Assemblage	fonction utilisateur
norm_Ad	EF/Assemblage	fonction utilisateur
norm_Ad2	EF/Assemblage	fonction utilisateur
norm_T	EF/Assemblage	fonction utilisateur
int_fonc_T	EF/Assemblage	fonction utilisateur

Calcul sur les tableaux de cellules

Nom	Sous rép. de EF	Parent(s)
init_transf_cell	mat_cell	fonction utilisateur
trans_sol_mat_sol_cell	mat_cell	init_transf_cell
trans_mat_mat_cell	mat_cell	EF_assemblage
		systeme_bord_cell
trans_mat_cell_mat	mat_cell	(")
supp_lin_col_vec_cell	mat_cell	EF_assemblage
supp_lin_col_mat_cell	mat_cell	EF_assemblage
add_mat_cell	mat_cell	EF_assemblage

4.1.1. LISTE DES REPERTOIRES ET FONCTIONS

Transformation des maillages créés par pdetool

Les fonctions `bmailP1`, `cmailP1`, `cmailP1b`, `dmailP1`, ci-dessous, sont utilisées pour créer des maillages avec des raffinements successifs. Elles utilisent (ou non) de manière diverse la fonction `jigglemesh` qui permet d'améliorer la qualité des maillages, mais dont l'utilisation conduit *a priori*, si on l'utilise à partir de l'éventuelle deuxième étape de raffinement, à des maillages non emboîtés. Pour regrouper des frontières entre elles (*cf.* Paragraphe 2.4), l'ordre choisi pour le regroupement doit coïncider avec l'orientation de ces frontières ce qui n'est pas toujours possible, notamment lorsque deux frontières contigües que l'on veut regrouper ont des orientations opposées. D'autre part, lorsque des morceaux de frontière du maillage ne comportent pas de sommets du maillage en dehors de leur extrémités, la fonction `bmailP1` peut ne pas prendre en compte ce maillage. Dans ces deux cas on utilisera les fonctions `bmailP1_b`, `cmailP1_b`, `cmailP1b_b`, `dmailP1_b`. Pour ces fonctions, le résultat sur la variable de sortie `Front` peut changer par rapport à `bmailP1`. L'utilisation de ces fonctions est décrite au Paragraphe 2.4.

Nom Sous rép. de EF Parent(s)

Utilisation de jigglemesh uniquement sur le premier maillage, pour emboîtement :

`bmailP1` `trait_maillage` fonction utilisateur

Variante 1. Utilisation de jigglemesh uniquement sur le dernier maillage :

`cmailP1` `trait_maillage` fonction utilisateur

Variante 2. Pas d'utilisation de jigglemesh, pour emboîtement :

`cmailP1b` `trait_maillage` fonction utilisateur

Variante 3. Utilisation de jigglemesh à chaque raffinement de maillage :

`dmailP1` `trait_maillage` fonction utilisateur

Autres versions : n'imposent pas de contraintes d'orientation des frontières pour leur regroupement éventuel :

`taille_maillage` `trait_maillage` fonction utilisateur

`bmailP1_b` `trait_maillage` fonction utilisateur

`bmailP1b_b` `trait_maillage` fonction utilisateur

`cmailP1b_b` `trait_maillage` fonction utilisateur

`cmailP1b_b` `trait_maillage` fonction utilisateur

Autres fonctions de maillage :

`export_mail` `trait_maillage` `bmailP1`, `cmailP1`,
 `cmailP1b`, `dmailP1`

`front_ordr` `trait_maillage` `bmailP1`, `cmailP1`,

		cmailP1b, dmailP1
taille_maillage	trait_maillage	fonction utilisateur
front_ordr_b	trait_maillage	bmailP1_b, cmailP1_b, cmailP1b_b, dmailP1_b

Raffinement adaptatif.

Le répertoire suivant permet de traiter le raffinement adaptatif, avec une fonction de maillage et des fonctions de calcul d'erreur. L'utilisation de ces fonctions est décrite au Paragraphe 2.11.

Nom	Sous rép. de EF	Parent(s)

Fonction de raffinement de maillage :

bmailP1_raf	adapt	fonction utilisateur
bmailP1_b_raf	adapt	fonction utilisateur

Fonctions pour le calcul d'estimateurs locaux :

norm_T_adapt	adapt	fonction utilisateur
norm_T_adapt_m	adapt	(")
norm_A_adapt	adapt	(")
norm_Ad_adapt	adapt	(")

Fonctions de base prenant en compte des dérivations d'ordre 2 :

F_base_adapt	adapt	F_base_T_adapt, F_base_A_adapt
F_base_T_adapt	adapt	pre_assemble_T_adapt,
F_base_A_adapt	adapt	pre_assemble_A_adapt, valeur_fonc_quad_K

Autres fonctions (en particulier valeur de la solution aux points de la formule de quadrature) :

pre_assemble_T_adapt	adapt	norm_T_adapt, norm_T_adapt_m valeur_fonc_quad_K
pre_assemble_A_adapt	adapt	norm_A_adapt, norm_Ad_adapt

Eléments Finis courbes

Les éléments courbes sont développés dans le répertoire `element_t`. Une description de l'utilisation de ces fonctions est donnée au Paragraphe 3.6.

Les fonctions ci-dessous avec le suffixe `_t` sont les versions éléments courbes des fonctions précédentes.

L'approximation par EF courbes est ici caractérisée par la donnée de la fonction $F_C \in [P_k^c]^2$ du Paragraphe 3.6, k étant le degré à choisir, la fonction F_C étant une bijection qui transforme le domaine initial à frontière polygonal en domaine à frontière courbe en laissant invariantes les arêtes intérieures.

4.1.1. LISTE DES REPERTOIRES ET FONCTIONS

Pour un maillage donné l'utilisateur aura à créer une fonction qui déplace les *ddl* frontières du P_k sur la nouvelle frontière courbe (*cf.* 3.6).

La fonction `EF_PrepA_t` remplace la fonction `EF_PrepA` et se différencie par l'appel de `EF_prepa_don_t` au lieu de `EF_prepa_don`.

Pour le calcul des *ddl* de l'espace d'EF qui sont l'image par la transformation F_C des *ddl* sur le maillage de départ, la fonction de préparation `EF_prepa_don_t` appelle les fonctions suivantes :

`assemble_coeff_T`, la fonction utilisateur décrivant F_C comme la fonction `mail-
lage1_t` dans le premier exemple du Paragraphe 3.6.1, `pre_assemble_T_t`.

La fonction d'assemblage `EF_assemblage_t` appelle les fonctions

`matrice_elem_Ad_t`, `matrice_elem_Ad_scm_t`, `matrice_elem_A_t`,
`matrice_elem_A_scm_t`, `matrice_elem_T_t`, `matrice_elem_t_scm_t`.

Les fonctions `pre_assemble_T_t` et `assemble_coeff_T` sont utilisées par :

`matrice_elem_T_t`, `matrice_elem_T_scm_t`, `matrice_elem_prepa_T_t`

et comme indiqué ci-dessus par `EF_prepa_don_t`. La fonction de tracé `trac_solEd_t` utilise aussi `pre_assemble_T_t`.

Les fonctions `pre_assemble_A_t` et `assemble_coeff_A` sont utilisées par :

`matrice_elem_A_t`, `matrice_elem_A_scm_t`, `matrice_elem_prepa_A_t`

ainsi que par

`matrice_elem_Ad_t`, `matrice_elem_Ad_scm_t`, `matrice_elem_prepa_Ad_t`.

Les fonctions donnant les formules de quadrature sont identiques : ce sont `assemble_coeff_T` et `assemble_coeff_T`, elle ne dépendent pas de F_C .

Le calcul des fonctions de base, de leur dérivées, de la transformation F_C et de sa matrice jacobienne sur les triangles ou les arêtes se fait par l'intermédiaire des fonctions `pre_assemble_T_t` et `pre_assemble_A_t` qui appellent respectivement les fonctions `F_base_T_t` et `F_base_A_t` qui elles mêmes appellent la fonction `F_base` de calcul de fonctions de base. La fonction `F_base` ne dépend pas de F_C .

Pour un triangle K donnée, la fonction `F_base_T_t_inv` permet le calcul de la réciproque par F_C d'un point de $F_C(K)$. Si ce point n'est pas dans $F_C(K)$, alors c'est l'image réciproque par la fonction notée F_C^K définie au Paragraphe (voir aussi 3.6), qui, lorsqu'elle existe, est calculée.

La fonction d'assemblage `EF_assemblage_t` appelle de manière optionnelle la fonction `changt_base_nt_t` qui appelle elle même la fonction `base_nt_t`, pour tenir compte des normales du domaine $F_C(\Omega)$ dans le cas de conditions imposées au bord sur les composantes normales ou tangentielles d'un champ de vecteur. Sinon c'est la fonction la fonction `changt_base_nt` appelant elle même la fonction `base_nt`, qui est utilisée, les normales choisies sont alors celles du maillage initial (non courbe), avec une moyenne pondérée pour les normales sur des sommets anguleux

du maillage. Un exemple d'utilisation de cette option est donnée au Paragraphe 3.6.2, dans la commande

```
cd C :/EF_text/exemples/exemple_stokes_courb2, instruct1_adapt
```

avec la syntaxe

```
[S,f] = EF_assemblage_t(donnees, Uc, 'o');
```

avec 'o' variable optionnelle pour l'appel de `changt_base_nt`.

Eléments courbes :

Nom

EF_Prepa_t

EF_prepa_don_t

EF_assemblage_t

pre_assemble_T_t

pre_assemble_A_t

pre_assemble_Tadapt_t

pre_assemble_Aadapt_t

matrice_elem_T_t

matrice_elem_A_t

matrice_elem_Ad_scm_t (en développement, parent EF_assemblage_t)

matrice_elem_Ad_t (en développement, parent EF_assemblage_t)

matrice_elem_T_scm_t

matrice_elem_A_scm_t

matrice_elem_prepa_T_t

matrice_elem_prepa_A_t

matrice_elem_prepa_Ad_t

F_base_T_t

F_base_A_t

F_base_T_t_inv (calcul de la réciproque de F_C)

F_base_Tadapt_t

F_base_Aadapt_t

F_base_adapt

norm_A_t (dérivées jusqu'à l'ordre un)

norm_T_t (dérivées jusqu'à l'ordre un)

norm_Tadapt_t (dérivées jusqu'à l'ordre deux)

norm_Tadapt_m_t (dérivées jusqu'à l'ordre deux)

norm_Aadapt_t (dérivées jusqu'à l'ordre deux)

norm_Adadapt_t (dérivées jusqu'à l'ordre deux)

int_fonc_T_t(calcul d'intégrale jusqu'à l'ordre un)

ddl_t (calcul des ddl de l'espace EF courbe)

base_nt_t (normale au domaine courbe)

changt_base_nt_t (normale au domaine courbe)

trac_solEd_t (tracé domaine courbe)

4.1.2. ALGORITHME GENERAL DU CODE

`trac_sol_coup_t` (coupe domaine courbe)
`valeur_fonc_EF_t` (valeur d'une fonction en un point de $F_C(K)$,
avec K triangle)
`pt_tri_t` (appartenance d'un point à $F_C(K)$, avec K triangle)

Fonctions utilisateurs : exemple des fonctions utilisées par la commande :

`cd C :/EF_text/exemples/exemple1_Laplacien, instruct1`

Fonctions utilisateurs appelées par `instruct1` :

Nom	Répertoire de C :/EF_text	Parent(s)
<code>instruct1</code>	<code>exemples/exemple1_Laplacien</code>	
<code>don1</code>	<code>exemples/exemple1_Laplacien</code>	<code>instruct1</code>
<code>eq_dom</code>	<code>exemples/exemple1_Laplacien</code>	<code>matrice_elem_t</code>
<code>scm_dom</code>	<code>exemples/exemple1_Laplacien</code>	<code>matrice_elem_t_scm</code>
<code>cb_don</code>	<code>exemples/exemple1_Laplacien</code>	<code>EF_assemblage</code>
<code>f3</code>	<code>exemples/exemple1_Laplacien</code>	sous fonction de <code>cb_don</code>

Ces fonctions concernent l'initialisation des données, des deux membres de l'équation et des conditions aux bord.

Fonctions du code appelées par `instruct1` :

`EF_prepa`
`EF_assemblage`
`trac_solEc`
`init_transf_cell`

Ces fonctions concernent le pré-assemblage, l'assemblage, le tracé des solutions, l'initialisation à 0 de l'itéré courant (à effectuer même pour les problèmes linéaires).

4.1.2. Algorithme général du code.

Nous donnons ci-dessous une présentation algorithmique de l'utilisation des fonctions du code pour sa partie principale qui est celle de l'assemblage. La partie du code consacrée au traitement des solutions (interpolation, graphisme) est traitée au Paragraphe 4.3 (description des algorithmes) ainsi qu'au Paragraphe 2.5 qui précède (utilisation des fonctions). La partie du code consacrée au maillage est traitée au Paragraphe 2.4 précédant, elle porte sur l'utilisation des fonctions de traitement de maillages obtenus par `pdttool`, la partie algorithmique n'est pas développée. Les autres parties du code sont consacrées à la résolution adaptative et aux EF courbes. Pour alléger le propos, nous avons omis la description algorithmique de ces parties qui est proche de celle du reste code. On se reportera aux paragraphes précédents 2.11 et 3.6 pour l'utilisation des fonctions concernées. Les numéros entre crochets ci-dessous indiquent la hiérarchie des instructions.

Algorithme : Code. Partie assemblage de système.

- [1] Ajout des répertoires du code au `path` :

`path_EF`

- Initialisation de la variable `donnees` décrivant le problème (Paragraphe 2.2) :

Nom des fichiers de données (Paragraphe 2.2.1).

Description des espaces d'EF (début du Paragraphe 2.2, Paragraphe 2.2.1 et aussi 2.3)

Description du maillage (Paragraphe 2.2.2 et aussi Paragraphe 2.4).

Description des formes linéaires et bilinéaires, conditions au bord, méthode de Lesaint-Raviart (Paragraphe 2.2.4)

- [1] Calculs de données relatives aux espaces d'EF et aux formules de quadrature (Variables du Paragraphe 2.2.3) :

`EF_prepa`

Syntaxe :

`EF_Prep(donnees) ;`

Avec en entrée `donnees` la variable de type `struct` décrivant les données, en sortie le fichier de type `.mat` relatif aux types d'espaces d'EF et au choix des formules de quadrature.

- [2] Sous-fonction faisant les calculs

`EF_prepa_don`

- [3] Calculs de valeurs de fonctions de base, de matrices d'assemblage de type norme d'énergie et graphe des matrices d'assemblage, boucle sur triangles et arêtes :

- [3] Calcul sur un triangle :

`matrice_elem_prepa_T`

- [4] Pré-calculs de matrices élémentaires (*cf.* Paragraphe 3.1.2.4)

`pre_assemble_T`

- [5] Fonctions de base (triangles)

`F_base_T`

- [6] Fonctions de base (*cf.* Paragraphe 2.3.3)

`F_base`

- [4] Formules de quadrature sur triangles (*cf.* Paragraphe 3.3.1, 3.3.3)

`assemble_coeff_T`

- [3] Calcul sur une arête frontière :

`matrice_elem_prepa_A`

4.1.2. ALGORITHME GENERAL DU CODE

- [4] Pré-calculs de matrices élémentaires (*cf.* Paragraphe 3.1.2.4)

`pre_assemble_A`

- [5] Fonctions de base (arêtes)

`F_base_A`

- [6] Fonctions de base (*cf.* Paragraphe 2.3.3)

`F_base`

- [4] Formules de quadrature sur arêtes (*cf.* Paragraphe 3.3.2, 3.3.3)

`assemble_coeff_A`

- [3] Calcul sur une arête :

`matrice_elem_prepa_Ad`

Mêmes fonctions appelées que dans le cas précédent

- [3] Numérotation et liste des arêtes du maillage :

`deg_lib/maillage_triangle_arete_ps`

- [4] Numérotation des arêtes :

`deg_lib/maillage_triangle_arete`

- [5] Tableau de connectivité des arêtes du maillage

`deg_lib/triangle_arete`

- [3] Numérotation et calcul des coordonnées des *ddl* des espaces d'EF (*cf.* Paragraphe 1.4 et Paragraphe 2.3.2) :

`deg_lib/maillage_ddl`

-
- [1] Initialisation de vecteurs de type cellules :

`init_transf_cell`

Syntaxe :

`Uc = init_transf_cell(rep, U, donnees, varargin) ;`

Essentiellement utilisée sous la forme

`Uc = init_transf_cell('init', [], donnees) ;`

pour initialiser `Uc` à la fonction nulle. Avec cette syntaxe, le fichier de nom `donnees.nom_prep` doit être déjà calculé avec `EF_Prep` : il donne la dimension des espaces d'EF composant l'espace EF produit E , cette dimension est nécessaire pour l'initialisation de `Uc`.

-
- [1] Assemblage du système

`EF_assemblage`

Syntaxe :

`[S,F] = EF_assemblage(donnees, Uc) ;`

Avec `donnees` variable de type `struct` décrivant les données, `Uc` variable de type `cell` décrivant un éventuel itéré ou la fonction nulle. En sortie, la matrice et le second membre du système de type `sparse`.

- [2] Boucles et calculs des matrices élémentaires du système dont :
- [2] Boucle sur les triangles pour l'assemblage de la forme bilinéaire (*cf.* Paragraphe 3.1.3.3) :

`matrice_elem_T` (forme bilinéaire)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2 et l'exemple du Paragraphe 2.2.4.1) :

`donnees.feq_Dom`

- [3] Pré-calculs de matrices élémentaires (*cf.* Paragraphe 3.1.2.4)

`pre_assemble_T`

- [3] Assemblage de systèmes (*cf.* Paragraphe 3.1.3.3)

`assemble_equ`

- [4] Sous-fonction auxiliaire. Accélération des calculs.

`assemble_equ/mult_cp_prep`

- [4] Fonction d'assemblage scalaire (*cf.* Paragraphe 3.1.2.4)

`equationJ`

- [5] Sous-fonction auxiliaire (accélération des calculs) :

`equationJ/mult_cp`

- [3] Formules de quadrature sur triangles (*cf.* Paragraphe 3.3.1, 3.3.3)

`assemble_coeff_T`

- [2] Boucle sur les triangles pour l'assemblage de la forme linéaire (*cf.* Paragraphe 3.1.3.3) :

`matrice_elem_T_scm` (second membre)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2 et l'exemple du Paragraphe 2.2.4.2) :

`donnees.fscm_Dom`

Mêmes fonctions appelées que pour `matrice_elem_T`

- [2] Boucle sur les arêtes frontières pour l'assemblage de la forme bilinéaire (*cf.* Paragraphe 3.1.3.3)

`matrice_elem_A` (forme bilinéaire)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2) :

4.1.2. ALGORITHME GENERAL DU CODE

`donnees.feq_Fr`

- [3] Pré-calculs de matrices élémentaires (*cf.* Paragraphe 3.1.2.4)

`pre_assemble_A`

- [3] Assemblage de systèmes (*cf.* Paragraphe 3.1.3.3)

`assemble_equ`

- [4] Sous-fonction auxiliaire. Accélération des calculs.

`assemble_equ/mult_cp_prep`

- [4] Fonction d'assemblage scalaire (*cf.* Paragraphe 3.1.2.4)

`equationJ`

- [5] Sous-fonction auxiliaire (accélération des calculs) :

`equationJ/mult_cp`

- [3] Formules de quadrature sur arêtes (*cf.* Paragraphe 3.3.2, 3.3.3)

`assemble_coeff_A`

- [2] Boucle sur les arêtes frontières pour l'assemblage de la forme linéaire (*cf.* Paragraphe 3.1.3.3)

`matrice_elem_A_scm` (second membre)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2) :

`donnees.fscm_Fr`

Mêmes fonctions appelées que pour `matrice_elem_A`

- [2] Boucle sur les arêtes pour l'assemblage de la forme bilinéaire (*cf.* Paragraphe 3.1.3.3, en développement, fonctionne)

`matrice_elem_Ad` (forme bilinéaire, en développement, fonctionne)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2) :

`donnees.feq_Ad`

Mêmes fonctions appelées que pour `matrice_elem_A`

- [2] Boucle sur les arêtes pour l'assemblage de la forme linéaire (*cf.* Paragraphe 3.1.3.3, en développement, fonctionne)

`matrice_elem_Ad_scm` (forme linéaire, en développement, fonctionne)

- [3] Fonction de `handle` donné par (*cf.* Syntaxe au Paragraphe 2.2) :

`donnees.fscm_Ad`

Mêmes fonctions appelées que pour `matrice_elem_A`

- [2] Boucle sur les arêtes pour la méthode de Lesaint-Raviart : (*cf.* Paragraphe 1.2)

`matrice_elem_Ad_int_LR` (forme bilinéaire)

- [3] Fonctions de **handle** donnés par la première colonne de cellules de (*cf.* Syntaxe au Paragraphe 2.2 et Tableau 2.2.1-1d) :

`donnees.D.LR`

Les inconnues transportées étant donnés par la deuxième colonne de cellules `donnees.D.LR`

- [3] Pré-calculs de matrices élémentaires (*cf.* Paragraphe 3.1.2.4)

`pre_assemble_A`

- [3] Formules de quadrature (*cf.* Paragraphe 3.3.2, 3.3.3)

`assemble_coeff_A`

- [2] Assemblage du système à partir des matrices obtenues pour chacune des boucles :

- [2] Addition de matrices cellules

`add_mat_cell`

- [2] Transformations du système dont :

- [2] Traitement des conditions au bord (*cf.* Paragraphe 3.2) qui comprend :

- [2] Traitement de la fonction décrivant les conditions aux bords par la fonction du code (*cf.* Paragraphe 3.2.3)

`cb`

- [2] Le **handle** de la fonction décrivant les conditions au bord, les priorités pour les coins (*cf.* Syntaxe au Paragraphe 2.2, les tableaux 2.2.4-2a, b, c et l'exemple du Paragraphe 2.2.4.3) est :

`donnees.fcb`

- [2] Traitement des conditions au bord sur des composantes normales ou tangentielles de champs de vecteurs (*cf.* Tableau 2.2.4-2c et Paragraphe 3.2.2) dont :

- [2] Evaluation de la fonction décrivant les conditions au bord

`donnees.fcb`

- [2] Calcul de la matrice de changement de base

`changt_base_nt`

- [3] Description par arêtes des *ddl* sur la frontière (*cf.* Paragraphe 3.2 et Paragraphe 3.2.3)

`Front_ddl`

- [3] Calcul de normales pour des *ddl* sur la frontière (*cf.* Paragraphe 3.2.2 et Paragraphe 3.2.3)

`base_nt`

- [2] Transformation de matrice de cellules en matrices et réciproquement pour écrire le système dans la nouvelle base

4.2. PRINCIPALES VARIABLES DU CODE

`trans_mat_cell_mat`

`trans_mat_mat_cell`

- [2] Traitement des conditions au bord de type $AX = b$ (*cf.* Tableau 2.2.4-2a et 2.2.4-2b dont :

- [2] Calcul de la matrice de changement de base

`changt_base_a`

- [2] Transformation de matrice de cellules en matrices et réciproquement pour écrire le système dans la nouvelle base :

`trans_mat_cell_mat`

`trans_mat_mat_cell`

- [2] Fixation des valeurs des fonctions aux *ddl* sur lesquels il y a des conditions au bord et calcul du nouveau système et second membre

`systeme_bord_cell`

- [3] Mettre à 0 et 1 des coefficients de matrices (*cf.* Paragraphe 3.2 et Paragraphe 3.2.3)

`transf_mat1`

- [3] Transformation de matrice de cellules en matrices et réciproquement pour la transformation `transf_mat1` et les opérations matricielles sur la matrice de type *sparse*

`trans_mat_cell_mat`

`trans_mat_mat_cell`

- [2] Retour à la base de départ, transformation de matrice de cellules en matrices et réciproquement

`trans_mat_cell_mat`

`trans_mat_mat_cell`

- [2] Elimination d'inconnues dont :

- [2] Suppression de lignes et de colonne d'un vecteur cellules

`supp_lin_col_vec_cell`

- [2] Suppression de lignes et de colonne d'une matrice de cellules

`supp_lin_col_mat_cell`

- [2] Modification du système par addition de matrices cellules

`add_mat_cell`

- [2] Passage matrice de cellules à matrice de type *sparse* en sortie :

`trans_mat_cell_mat`

4.2. Principales variables utilisées par le code.

Nous donnons la description des principales variables utilisées dans le code. Leur définition peut changer dans certaines fonctions, comme par exemple la variable Uc dans `inv_interp_EF_ddl`, où les cellules de Uc représentent les valeurs aux ddl d'une fonction au lieu, comme habituellement, des valeurs des composantes attachées à ces ddl .

Variables diverses.

`donnees` : STRUCT ARRAY. Variable décrivant les données du problème EF.

`donnees.nom_prep` : CHAR ARRAY. Nom du fichier relatif au espaces d'EF et aux formules d'intégration (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

`donnees.nom_mail` : CHAR ARRAY. Nom du fichier au maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

`D` : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ D du Tableau 2.2.1-1c, Paragraphe 2.2.1.

Variable utilisée dans `maillage_ddl` et qui prend le nom e dans `base_T`, dans `base_A` :

`rep` : DOUBLE ARRAY. Numéro de l'espace d'EF choisi (*cf.* Paragraphe 1.4).

Pour $1 \leq i \leq 9$:

- `rep = i` désigne le P_i^c ,
- `rep = 10i` désigne le P_i^d (exemple, si $i = 1$, `10i` désigne le nombre 101),

Pour $1 \leq i \leq 9$ et $0 \leq j \leq 9$:

- `rep = ij` désigne le $P_{i,j}^c$ défini en (1.21) au Paragraphe 1.4,
- `rep = 1ij` désigne le $P_{i,j}^d$ défini en (1.22) au Paragraphe 1.4.

Le P_0^d est désigné par `rep = 100`, le P_{10}^c par `rep = 200` et le P_{10}^d par `rep = 300`.

La description de ces espaces d'EF est donnée au Paragraphe 1.4.

Variables du système EF.

`S` : DOUBLE ARRAY. Matrice finale du système, après assemblage sur l'espace EF produit courant E , après changement de base par rotations successives pour les conditions au bord portant sur les composantes normales ou tangentielles de champs de vecteurs, changement de base pour des conditions de type $AX = b$ sur le bord, transformation du système obtenu pour la prise en compte des conditions au bord (*cf.* (1.4)) et retour à la base initiale et enfin élimination d'inconnues du système : inconnues composantes (*cf.* (1.3)') de l'espace EF produit courant ou inconnues ponctuelles.

`f` : DOUBLE ARRAY. Représente le second membre final, tenant compte des conditions au bord et l'élimination d'inconnues.

4.2. PRINCIPALES VARIABLES DU CODE

Uc : **CELL ARRAY**. Vecteur colonne dont les cellules sont des vecteurs colonnes de type **DOUBLE ARRAY**. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (cf. Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques. Cette variable peut représenter d'autres valeurs comme dans la fonction `inv_interp_EF_ddl` où elle désigne les valeurs d'une fonction aux *ddl* de l'espace EF produit courant.

Variables de maillage.

T : **DOUBLE ARRAY**. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

X,Y : **DOUBLE ARRAY**. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Front : **CELL ARRAY**. Vecteur colonne. La cellule i de **Front** est un vecteur ligne de type **DOUBLE ARRAY** qui décrit consécutivement sur la frontière les numéros de sommets du morceau de frontière numéro i , cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Sur l'utilisation de Front. Cette variable décrit les sommets du maillage situés sur la frontière du domaine. Les sommets sont énumérés dans un ordre qui les décrit consécutivement sur la frontière.

Cet ordre intervient dans les fonctions du répertoire `trait_maillage` et dans la fonction `bmailP1_raf` du répertoire `adapt` qui permettent le passage des maillages données par `pdetool` à la structure de maillage utilisée dans le code. L'utilisation de **Front** se fait dans ces fonctions par l'intermédiaire de la fonction `front_ordr` du répertoire `trait_maillage`, cette fonction gérant les structures des sommets frontières.

Cet ordre intervient aussi dans la fonction `maillage_triangle_arete_ps` pour la construction des variables **Pa**, **Pfc** et **Pf** qui renvoient une description par arêtes des sommets du maillages.

Cependant cette ordre n'intervient pas dans l'assemblage proprement dit : dans les fonctions de traitement des conditions au bord **cb** et `changt_base_a`, seules interviennent les dimensions de **Front**, qui est une matrice de type **CELL ARRAY**, et les variable **Pfc** et **Arc** qui est une description ordonnée arête par arête des *ddl* (l'ordre intervient sur l'arête) mais avec un ordre qui n'intervient pas dans **cb** et `changt_base_a`.

Enfin la variable **Front** n'est pas utilisée que dans les fonctions ci-dessus, la fonction utilitaire `verif_don_EF` et dans les fonctions du répertoire `element_t` pour le traitement des EF courbes.

Lors de la création d'un nouveau maillage, par `pdetool` ou par un autre moyen, on pourra vérifier par précaution l'ordre des sommets de **Front**, ainsi que l'existence, dans le cas d'une frontière qui se referme, du même sommet à chaque extrémité du vecteur tiré de la cellule de **Front** qui décrit les sommets de cette frontière.

Variables liées à la géométrie du domaine.

A : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes du maillage) $\times$ 4. Matrice de correspondance entre les numéros d'arêtes et les numéros de triangles. $A(i, 1)$ est le numéro de triangle auquel appartient l'arête n° i , $A(i, 2)$ le numéro local de cette arête sur ce triangle. Même chose pour $A(i, 3)$ et $A(i, 4)$ si l'arête est portée par un second triangle, dans le cas contraire ces deux derniers termes prennent la valeur 0.

B : DOUBLE ARRAY. Matrice de taille (Nombre de triangles sur la frontière) $\times$ 3. Matrice de correspondance entre les triangles et les arêtes. $B(i, k)$ est le numéro d'arête dans la numérotation globale des arêtes de l'arête n° k du triangle i , dans la numérotation locale des arêtes de ce triangle ($k = 1, 2, 3$).

XA, **YA** : DOUBLE ARRAY. Vecteurs lignes des coordonnées cartésiennes des milieux d'arête. $XA(i)$, $YA(i)$, sont les coordonnées du point au milieu de l'arête n° i .

Am : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes du maillage) $\times$ 2. Description arête par arête des sommets du maillage : $Am(i, :)$ désigne les numéros des deux sommets de l'arête numéro i du maillage. L'arête n'est pas orientée. On a la relation $Am(i, :) = T(Pa(i, 1), z)$ avec $z = [1, 2]$ si $Pa(j, 2) = 1$, $z = [1, 3]$ si $Pa(j, 2) = 2$, et $z = [2, 3]$ si $Pa(j, 2) = 3$.

Pa : DOUBLE ARRAY. Matrice de taille (Nombre d'arête du maillage) $\times$ 5. Description des arêtes du maillage tel que $Pa(j, 1)$ est un numéro de triangle auquel appartient l'arête j et $Pa(j, 2)$ son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour $Pa(j, 3 : 4)$ si l'arête appartient à un deuxième triangle sinon $Pa(j, 3)$ et $Pa(j, 4)$ sont nuls (arête sur la frontière). Enfin $Pa(j, 5)$ est le numéro du morceau de frontière si l'arête j est sur la frontière, nul sinon ou si on a posé **Front** = []. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Pfc : CELL ARRAY. Vecteur colonne de cellules de longueur égale au nombre de morceaux de frontière du domaine. Cette variable a donc le même nombre de cellules que la variable de maillage **Front** (cf. Tableau 2.2.2-1, Paragraphe 2.2.2). Sur chaque morceau de frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur ce morceau de frontière. La cellule $Pfc\{i\}$ est un tableau de trois colonnes de type DOUBLE ARRAY qui décrit les arêtes du morceau de frontière numéro i de la manière suivante. Soit une arête de numéro j dans sa numérotation sur le morceau de frontière numéro i , alors $Pfc\{i\}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $Pfc\{i\}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3) et $Pfc\{i\}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes. **Pf** vaut [] si **Front** = []. Le calcul de cette variable est réalisé par `maillage_triangle_arete_ps` à partir de **Front** dans l'ordre des arêtes de **Front**{1}, **Front**{2} etc. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Pf : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes sur la frontière) $\times$ 4. Sur la frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur la frontière. Soit une arête de numéro j dans sa numérotation sur

4.2. PRINCIPALES VARIABLES DU CODE

le morceau la frontière, $\text{Pf}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $\text{Pf}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3), $\text{Pf}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes et $\text{Pf}(j, 4)$ le numéro de morceau de frontière auquel appartient cette arête. Pfc vaut $[]$ si $\text{Front} = []$. Les variables Pf et Pfc contiennent la même information si l'on sait que $\text{Pf}(:, 1 : 3) = [\text{Pfc}\{1\}; \dots; \text{Pfc}\{\text{end}\}]$. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables liées aux *ddl*.

Td : DOUBLE ARRAY. Matrice de taille (Nombre de triangles) $\times$ (Nombre de *ddl* par éléments). Tableau décrivant les *ddl* triangles par triangles. $\text{Td}(i, :)$ est le vecteur des numéros de *ddl* de l'EF désigné par **rep**. Dans le cas du P_0 discontinu, les *ddl* sont les centres de gravité des triangles et chaque *ddl* a le même numéro que le triangle auquel il appartient : $\text{Td}(i, :) = i$. Sinon, $\text{Td}(i, 1 : 3)$ est le vecteur des numéros des trois sommets du triangle i donné par $\text{T}(i, 1 : 3)$. Ensuite $\text{Td}(i, 4 : \text{end})$ indique les numéros des *ddl* situés sur les arêtes (hormis les sommets), puis à l'intérieur du triangle numéro i . Si a_1, a_2, a_3 sont les numéros de sommet d'un triangle i donnés dans leur ordre de numérotation locale sur ce triangle, $\text{Td}(i, 4 : \text{end})$ donne alors les numéros des *ddl* qui se succèdent dans l'ordre suivant : d'abord ceux de l'arête a_1a_2 en allant de a_1 vers a_2 (arête 1, localement), puis ceux de a_1a_3 en allant de a_1 vers a_3 (arête 2) puis ceux de a_2a_3 (arête 3) en allant de a_2 vers a_3 , puis les *ddl* intérieurs, ce en parcourant les droites parallèles à a_1a_2 , dans le sens a_1 vers a_2 et en partant de la droite la plus proche du segment a_1a_2 jusqu'à la plus éloignée.

Ad : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes) $\times$ (Nombre de *ddl* sur les arêtes, sommets compris) avec une seule colonne (nulle) s'il n'y pas de *ddl* sur les arêtes. Description des *ddl* arête par arête. Le terme $\text{Ad}(i, :)$ vaut 0 s'il n'y a pas de *ddl* d'arêtes comme pour les espaces d'EF discontinus. Sinon, $\text{Ad}(i, [1, 2])$ prend les valeurs des sommets de l'arête i (ces valeurs sont données par $\text{Am}(i, [1, 2])$, où Am est la variable de description des arêtes des sommets du maillage calculée par la fonction `maillage_triangle_arete_ps`). Ensuite, s'il y a des *ddl* sur cette arête privée de ses extrémités, $\text{Ad}(i, [3, \dots])$ prend les valeurs des numéros de ces *ddl* en les décrivant consécutivement du sommet d'arête $\text{Ad}(i, 1)$ vers le sommet d'arête $\text{Ad}(i, 2)$. Cette variable ne sert que dans la fonction **cb** pour les conditions de type Dirichtlet où l'ordre de numérotation des *ddl* n'est pas utilisé.

Xd, Yd : DOUBLE ARRAY. Vecteur ligne de taille égale au nombre de *ddl*. Coordonnées en x et y des *ddl*.

Xc, Yc : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs $\text{Xc}\{i\}$ et $\text{Yc}\{i\}$ sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice $\text{Tc}\{i\}$ décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Arc : CELL ARRAY. Vecteur ligne. $\text{Arc}\{i\}$ est une matrice de type DOUBLE ARRAY décrivant les *ddl* arêtes par arêtes du *i*-ème type d'espace d'EF choisi, cf. le Tableau 2.2.3-1b du Paragraphe 2.2.3, $\text{Arc}\{i\}$ est un vecteur colonne nul dans le cas d'un espace d'EF qui n'a pas de *ddl* d'arêtes comme les espaces d'EF discontinus de numéros 100 à 199 et de numéro 300. La valeur de $\text{Arc}\{i\}$ est donnée par la valeur de la variable **Ad** calculée dans **maillage_ddl** dans le cas du *i*-ème type d'espace d'EF choisi.

Dnc : DOUBLE ARRAY. Vecteur ligne. $\text{Dnc}(i)$ est la dimension du *i*-ème espace d'EF de l'espace EF produit *E*.

Ed : DOUBLE ARRAY. Vecteur ligne. $\text{Ed}(i)$ est la dimension du *i*-ème type d'espace d'EF choisi.

Variables contenant des matrices d'assemblage ou relative à ces matrices.

a) Variables ne dépendant que du type des espaces d'EF choisis.

Ass_Dom : CELL ARRAY. Tableau de cellules $N \times N_E \times 3$ où *N* est le nombre de triangles et $N_E = \text{length}(\text{donnees.D.E})$ est le nombre de type d'espace utilisés (cf. le début du Paragraphe 2.2). Le terme $\text{Ass\_Dom}\{i, j, k\}$ est le tableau des valeurs, aux points de la formule de quadrature sur le triangle *K* numéro *i*, de la dérivée d'ordre $k-1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par $\text{donnees.D.E}(j)$. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.D.intT** utilisée sur les triangles et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ **Ass_Dom** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Ass_Ad : CELL ARRAY. Tableau de cellules $N \times N_E \times 3 \times 2$ où *N* est le nombre d'arêtes et $N_E = \text{length}(\text{donnees.D.E})$. Le terme $\text{Ass\_Ad}\{i, j, k, m\}$ est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K_m de numéro $\text{Pa}(i, 1)$ pour $m = 1$ et $\text{Pa}(i, 3)$ pour $m = 2$ pour l'arête de numéro *i* dans la numérotation globale des arêtes, de la dérivée d'ordre $k-1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par $\text{donnees.D.E}(j)$. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Pour $m = 1$, on obtient le résultat concernant la restriction des fonctions de base sur le premier triangle auquel appartient l'arête *i* et de numéro donné par $\text{Pa}(i, 1)$, pour $m = 2$, le résultat concerne l'autre éventuel triangle de numéro $\text{Pa}(i, 3)$, s'il n'y a pas de second triangle on pose $\text{Ass\_Dom}\{i, j, k, m\} = []$. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.intA** utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ **Ass_Ad** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

4.2. PRINCIPALES VARIABLES DU CODE

Ass_Fr : CELL ARRAY. Tableau de cellules $N \times N_E \times 3$ où N est le nombre de d'arêtes sur la frontière et $N_E = \text{length}(\text{donnees.D.E})$. Sur la frontière, les arêtes reçoivent une numérotation locale, l'arête numéro i est décrite par $\text{Pf}(i, :)$. L'ordre choisi est celui des arêtes de **Front**{1}, **Front**{2} etc. (cf. aussi les variables **Pf** et **Pfc**). Le terme **Ass_Fr**{ i, j, k } est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K de numéro $\text{Pf}(i, 1)$ qui porte l'arête de numéro i dans la numérotation locale des arêtes sur la frontière, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par $\text{donnees.D.E}(j)$. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.intA** utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace d'EF sur le triangle, l'ordre des fonctions de base étant celui donné au Paragraphe 2.3.3. Cette variable représente le champ **Ass_Fr** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

SL_Dom : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par $\text{donnees.D.E}(k)$. Le terme **SL_Dom**{ i, j, k, m } est une matrice de type **SPARSE ARRAY** de dimension $\dim(F_i) \times \dim(F_j)$. La matrice **SL_Dom**{ i, j, k, m } est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par

$$\int_{\Omega} v_{,k-1} w_{,m-1}.$$

Le terme (p, q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$, où $\{u^k\}_k$ désigne la base de l'espace d'EF F_k . Les bases sont décrites au Paragraphe 1.4 et au Paragraphe 2.3.2, sur un maillage exemple. Cette variable représente le champ **SL_Dom** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Cette variable peut être utilisée pour assembler des systèmes ou pour des calculs de norme d'énergie comme dans la fonction **norm_energ_cell** ou **norm_bilin_cell**.

SL_Ad : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Analogue de **SL_Dom** relativement à la forme bilinéaire

$$\int_{\Gamma} v_{,k-1} w_{,m-1} + \frac{1}{2} \int_{\Sigma_1} v_{,k-1} w_{,m-1} + \frac{1}{2} \int_{\Sigma_2} v_{,k-1} w_{,m-1},$$

avec Σ_p , $p = 1$ ou 2 réunion des arêtes intérieures, $p = 1$ signifiant que, sur une arête i donnée, on considère la restriction de $v_{,k-1} w_{,m-1}$ au triangle de numéro **Pa**($i, 1$) du Tableau 2.2.3-1b et pour $p = 2$ au triangle de numéro **Pa**($i, 3$). Cette variable représente le champ **SL_Ad** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

SL_Fr : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Analogue de **SL_Dom** relativement à la forme bilinéaire

$$\int_{\Gamma} v_{,k-1} w_{,m-1},$$

où Γ est la frontière de Ω . Cette variable représente le champ **SL_Fr** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

f_Dom : CELL ARRAY. Matrice de taille $N_E \times N_E$ avec $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par $\text{donnees.D.E}(k)$, la variable $\text{f\_Dom}\{i, j\}$ est de type **SPARSE ARRAY** et de taille $\dim(F_i) \times \dim(F_j)$ et le terme $\text{f\_Dom}\{i, j\}(p, q)$ vaut 1 si l'intersection des supports de la p -ième fonction de base de F_i et de la q -ième fonction de base F_j est de mesure non nulle et $\text{f\_Dom}\{i, j\}(p, q) = 0$ sinon.

F_Fr : CELL ARRAY. Analogue de **f_Dom**, la mesure est ici celle des arêtes frontières.

F_Ad : CELL ARRAY. Analogue de **f_Dom**, la mesure est ici celle des arêtes du domaine.

Nalloc_Dom : DOUBLE ARRAY. Matrice de taille $N_E \times N_E$, avec $N_E = \text{length}(\text{donnees.D.E})$. Le terme $\text{Nalloc\_Dom}(i, j)$ donne une estimation de la taille mémoire de la matrice d'assemblage de la forme bilinéaire $\int_{\Omega} uv$ définie sur l'espace $F_j \times F_i$ ou F_k est l'espace d'EF dont le numéro est donné par $\text{donnees.D.E}(k)$.

Nalloc_Ad : DOUBLE ARRAY. Matrice de taille $N_E \times N_E$, avec $N_E = \text{length}(\text{donnees.D.E})$. Analogue de **Nalloc_Dom** pour la forme $\int_{\Sigma} uv$.

Nalloc_Fr : DOUBLE ARRAY. Matrice de taille $N_E \times N_E$, avec $N_E = \text{length}(\text{donnees.D.E})$. Analogue de **Nalloc_Dom** pour la forme $\int_{\Gamma} uv$.

b) Matrices d'assemblage et second membres EF relatives à l'espace produit.

Sc_Dom : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Représentation de la matrice par blocs correspondant à l'assemblage de la partie en Ω de la forme bilinéaire $\mathcal{A}$ (terme en a_{Ω}) de (1.5), Paragraphe 1.2. La forme bilinéaire est définie par la fonction désignée par le champ **donnees.feq_Dom** (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage se fait par boucle sur les triangles, le rôle de la fonction **matrice_elem_T** est de calculer la contribution du triangle courant à la forme bilinéaire. Le terme $\text{Sc\_Dom}\{i, j\}$ est la matrice de taille $\dim(E_i) \times \dim(E_j)$ qui correspond à la matrice d'assemblage de la restriction de la partie de la forme bilinéaire qui porte sur $E_j \times E_i$, E_k désignant le k -ième espace de l'espace EF produit E et de numéro d'espace $\text{donnees.D.E}(\text{donnees.D.f}(k))$.

Sc_Fr : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Représentation de la matrice par blocs correspondant à l'assemblage de la partie en Γ de la forme bilinéaire $\mathcal{A}$ (terme en a_{Γ}) de (1.5), Paragraphe 1.2. Analogue de **Sc_Dom**, ici, pour l'assemblage de la partie en Γ au lieu de Ω . La forme bilinéaire est définie par le champ **donnees.feq_Fr**. L'assemblage se fait par boucle sur les arêtes de la frontière, le rôle de la fonction **matrice_elem_A** est de calculer la contribution de l'arête frontière courante à la forme bilinéaire. Le terme $\text{Sc\_Fr}\{i, j\}$ est la matrice de taille $\dim(E_i) \times \dim(E_j)$ qui correspond à la matrice d'assemblage de la restriction de la partie de la forme bilinéaire qui porte

4.2. PRINCIPALES VARIABLES DU CODE

sur $E_j \times E_i$, E_k désignant le k -ième espace de l'espace EF produit E et de numéro d'espace `donnees.D.E(donnees.D.f(k))`.

Sc_Ad : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Analogue de **Sc_Fr** pour l'assemblage dans la forme bilinéaire $\mathcal{A}$ de (1.5), Paragraphe 1.2, de la partie en $\bar{\Sigma}$ (réunion de toutes les arêtes du maillage) avec l'intégration du terme en $a_{\bar{\Sigma}}$. A ce terme s'ajoute la contribution du terme en a_{LR} de la méthode de Lesaint-Raviart. La forme bilinéaire $\bar{\Sigma}$ est définie par le champ `donnees.feq_Ad` et l'assemblage se fait par la fonction `matrice_elem_Ad`. Pour la méthode de Lesaint-Raviart, la forme bilinéaire est définie par le champ `donnees.D.LR` et l'assemblage se fait par la fonction `matrice_elem_Ad_int_LR`. La partie en $a_{\bar{\Sigma}}$ pourra être modifiée par l'utilisateur suivant le problème à traiter, ceci par le biais d'une fonction analogue à celle de `matrice_elem_Ad` à insérer dans une boucle du même type que celle effectuée sur les arêtes dans la fonction `EF_assemblage`.

Fc_Dom_scm : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Représente la matrice par blocs de l'assemblage de la partie en Ω de la forme bilinéaire $\mathcal{B}$ (terme en b_{Ω}) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Dom` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les triangles avec utilisation de la fonction `matrice_elem_T_scm`. Le terme `Fc_Dom_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`.

Fc_Fr_scm : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Analogue de **Fc_Dom_scm** relativement à la partie sur Γ de la forme bilinéaire $\mathcal{B}$. Représente la matrice par blocs de l'assemblage de la partie en Γ de la forme bilinéaire $\mathcal{B}$ (terme en b_{Γ}) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Fr` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les arêtes frontières avec utilisation de la fonction `matrice_elem_A_scm`. Le terme `Fc_Fr_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`.

Fc_Ad_scm : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Analogue de **Fc_Dom_scm** relativement à la partie sur $\bar{\Sigma}$ (réunion de toutes les arêtes du maillage) de la forme bilinéaire $\mathcal{B}$. Représente la matrice par blocs de l'assemblage de la partie en $\bar{\Sigma}$ de la forme bilinéaire $\mathcal{B}$ (terme en $b_{\bar{\Sigma}}$) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Ad` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les arêtes du maillage avec utilisation de la fonction `matrice_elem_Ad_scm`. Le terme `Fc_Ad_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`. En développement,

fonctionne.

Handles sur les fonctions définissant l'EDP et les conditions au bord.

Mat_syst_dom : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les EDP du premier membre sur Ω et qui correspond au terme a_Ω qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feq_Dom** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **Mat_syst_dom** = **donnees.feq_Dom**.

Mat_syst_Fr : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur Γ et qui correspond au terme a_Γ qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feq_Fr** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_syst_Fr** = **donnees.feq_Fr**.

Mat_syst_Ad : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur $\bar{\Sigma}$ et qui correspond au terme $a_{\bar{\Sigma}}$ qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feq_Ad** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_syst_Ad** = **donnees.feq_Ad**. En développement, fonctionne.

Mat_syst_Ad_int_LR : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur $\bar{\Sigma}$ et qui correspond au terme a_{LR} qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). La fonction définit alors le champ de vecteur de $\mathbb{R}^2$ suivant lequel on dérive. En pratique cette fonction est désignée par la variable **D_LR** décrite au Tableau 2.2.1-1d du Paragraphe 2.2.1. Les n_{LR} fonctions de (1.17) sont données successivement par **donnees.D_LR**{1, 1}, **donnees.D_LR**{2, 1}, ..., **donnees.D_LR**{ n_{LR} , 1}. La syntaxe de la fonction désignée par **Mat_syst_Ad_int_LR** est celle de la fonction donnée au Paragraphe 2.2.4 et au Tableau 2.2.4-1 de ce Paragraphe (cf. le cas de la méthode de Lesaint-Raviart).

Mat_syst_fscm_Dom : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit le second membre sur Ω et qui correspond au terme b_Ω qui intervient dans la forme linéaire $\mathcal{B}$ de (1.10). Ce **handle** correspond à la valeur du champ **fscm_Dom** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_syst_fscm_Dom** = **donnees.fscm_Dom**.

Mat_syst_fscm_Fr : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit le second membre sur Γ et qui correspond au terme b_Γ qui intervient dans la forme linéaire $\mathcal{B}$ de (1.10). Ce **handle** correspond à la valeur du champ **feq_fscm_Fr** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **Mat_syst_fscm_Fr** = **donnees.feq_fscm_Fr**.

Mat_syst_fscm_Ad : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit le second membre sur $\bar{\Sigma}$ et qui correspond au terme $b_{\bar{\Sigma}}$ qui intervient dans la forme linéaire $\mathcal{B}$ de (1.10). Ce **handle** correspond à la valeur du champ **feq_fscm_Ad** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_fscm_Ad** = **donnees.feq_fscm_Ad**. En développement, fonctionne.

4.2. PRINCIPALES VARIABLES DU CODE

fc : **FUNCTION_HANDLE** **ARRAY**. Handle sur la fonction qui permet de définir les conditions au bord de type (1.4) et ses variantes. Ce **handle** correspond à la valeur du champ **donnees.fc** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **fc = donnees.fc**.

Variables liées au tracés (répertoire **trace_graphe_interp**).

r : **DOUBLE** **ARRAY**. Variable portant sur les composantes de l'espace EF produit E . Vecteur ligne ou colonne **r**(i) désigne la composante de la fonction à, suivant le cas, interpoler, projeter, tracer.

M : **DOUBLE** **ARRAY**. Variable de raffinement de maillage pour l'interpolation et le tracé de fonctions de l'espace EF produit E . Chaque triangle du maillage est subdivisé en M^2 sous-triangles égaux. L'ordre de numérotation des triangles et des sommets est donné pour le cas $M = 4$ sur la figure 4.2-1 ci-dessous.

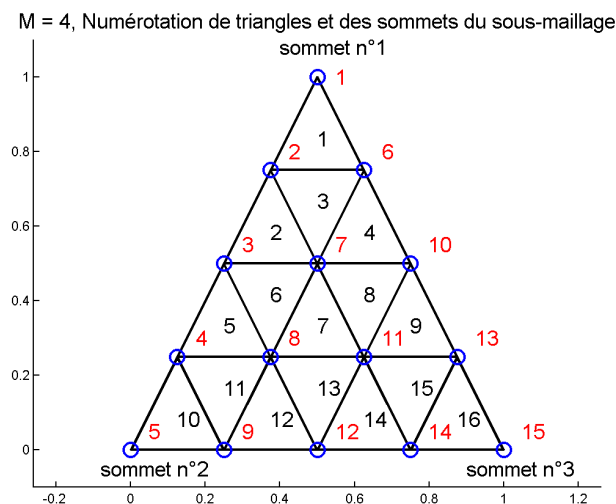


Figure 4.2-1

Z : **DOUBLE** **ARRAY**. Matrice de $\dim(E_K) \times \text{length}(L1)$ des valeurs des fonctions de base d'un espace d'EF E_K sur K , telle que **Z**(i, j) est la valeur de la dérivée partielle **der** = 0, 1 ou 2 de la i -ème fonction de base de E_K au point de coordonnées barycentriques **L1**(j), **L2**(j) et **L3**(j).

der : **DOUBLE** **ARRAY**. Les valeurs de **der** = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

nfig : **DOUBLE** **ARRAY** ou **STRUCT** **ARRAY**. Variable de figure. Lorsque **nfig** est de type **DOUBLE** **ARRAY**, la composante **r**(i) de la fonction à tracer est tracée sur la figure numéro i si **nfig** = [] et sur la figure numéro **nfig**-1+ i si **nfig** est un entier. Lorsque **nfig** est de type **STRUCT** **ARRAY**, **nfig.n** indique le numéro de figure sur lesquelles seront tracés tous les graphes dans un tableau de **nfig.t**(1) $\times$ **nfig.t**(2) figures.

vec : DOUBLE ARRAY. Variable pour les tracés de champs de vecteur. Matrice de deux colonnes. Pour le tracé du champs de vecteurs donnés par les composantes $\text{vec}(k, 1)$ et $\text{vec}(k, 2)$ de la fonction représentée par Uc . Peut prendre la valeur [].

X2, Y2 : DOUBLE ARRAY. Coordonnées des sommets du maillage obtenu par raffinement du maillage de l'espace EF produit courant E . Le maillage raffiné n'est pas de type EF. Le maillage de chaque triangle en sous-triangles est de type maillage EF.

T2 : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage raffiné. Les numéros de sommets du triangle k vont de $1 + (k - 1)(M + 2)(M + 1)/2$ à $k(M + 2)(M + 1)/2$, cette numérotation n'est pas de type maillage EF.

U : CELL ARRAY. Vecteur colonne de même taille que U . Le terme $\text{U}\{i\}$ est soit vide, soit, lorsqu'il a été calculé, le vecteur colonne telle que $\text{U}\{i\}(j)$ est la valeur au point $(\text{X2}(i), \text{Y2}(i))$ de la i -ème composante de la fonction (ou de l'une de ses dérivées, si $\text{der} = 1$ ou 2) représentée par Uc . Cette variable peut représenter d'autres valeurs comme Uc ci-dessus.

P : CELL ARRAY. On suppose que l'on calcule le polynôme d'interpolation de chaque fonction représentée par $\text{Uc}\{r_1\}, \dots, \text{Uc}\{r_p\}$ avec $r_1 < \dots < r_i < \dots < r_p$ donné. P est un vecteur ligne de longueur m_r .

- Si $i \in \{r_1, \dots, r_p\}$, $\text{P}\{i\}(k, :)$ est le vecteur ligne de type DOUBLE ARRAY des coefficients du polynôme $p^{i,k}$ de degré inférieur ou égal à M qui interpole la fonction représentée par $\text{Uc}\{i\}$ aux $(M + 1)(M + 2)/2$ sommets des triangles obtenus par division en M^2 triangles égaux du triangle numéro k du maillage (cf. la Figure 4.2-1, Paragraphe 4.2). Les coefficients sont les composantes de $p^{i,k}$ dans la base suivante :

$$1, x, x^2, \dots, x^M, y, xy, x^2y, \dots, x^{M-1}y, \dots, y^{M-1}, xy^{M-1}, y^M.$$

- Sinon, on a $\text{P}\{i\} = []$.

Q : CELL ARRAY. Vecteur ligne de même longueur que P , on reprend les notations utilisées pour la variable P .

- Si $i \in \{r_1, \dots, r_p\}$, $\text{Q}\{i\}\{k\}$ est une matrice carrée de type DOUBLE ARRAY de taille $M + 1$ qui représente le polynôme $p^{i,k}$ donné dans la définition de la variable P ci-dessus. Pour le triangle numéro k , le coefficient de $x^{m-1}y^{n-1}$ de $p^{i,k}$, $2 \leq m + n \leq M + 2$, est donné par le terme (m, n) de $\text{Q}\{i\}\{k\}$. Si $m + n > M + 2$ on pose $\text{Q}\{i\}\{k\} = 0$ (les termes sous l'anti-diagonale de $\text{Q}\{i\}\{k\}$ sont nuls). La valeur de $p^{i,k}$ en un point (x, y) est donnée par

$$p^{i,k}(x, y) = \text{sum}(\text{sum}(\text{Q}\{i\}\{k\} * (\text{x}.\wedge[0 : M]' * \text{y}.\wedge[0 : M]))).$$

- Sinon, on a $\text{Q}\{i\} = []$.

QX, QY : CELL ARRAY. Analogues de Q : $\text{QX}\{i\}\{k\}$ et $\text{QY}\{i\}\{k\}$ sont les deux matrices carrées de type DOUBLE ARRAY de taille $M + 1$ qui représentent respectivement les polynômes $p_{x,x}^{i,k}$ et $p_{y,y}^{i,k}$ de la définition de P ci-dessus avec la même représentation que Q ci-dessus.

4.2. PRINCIPALES VARIABLES DU CODE

Variables liées à l'assemblage élémentaire (répertoire Assemblage_el-ementaire).

omega : DOUBLE ARRAY.

Pour **cas** = 2 : Vecteur ligne ou colonne de taille $\dim(E_K)$ où E_K est l'espace d'EF sur un triangle K , de numéro d'espace d'EF **espace**. Ce vecteur représente alors les composantes d'une fonction dans la base de l'espace d'EF désigné par la variable **espace** décrite ci-après.

Pour **cas** = 1, 2 ou 4 : le résultat donné par la fonction est indépendant de **omega** qui peut être mis à [].

espace : DOUBLE ARRAY. Numéro d'espace d'EF sur un triangle K . Ces espaces sont décrits au Paragraphe 1.4. Les numéros d'espaces sont actuellement :

1 à 9 : éléments P_1 à P_9 continus, 200 : P_{10} continu, 10 à 99 : éléments continus à «bulles», exemples :

13 : $P_1 \oplus$ bulle P_3 ,

14 : $P_1 \oplus 3$ bulles P_4 ,

23 : $P_1 \oplus$ bulle P_3 ,

24 : $P_3 \oplus 3$ bulles P_4 somme non directe.

Ensuite, 100 à 109 : éléments P_0 à P_9 discontinus, 300 : P_{10} discontinu, 110 à 199 : éléments discontinus à «bulles», exemple :

113 : $P_1 \oplus$ bulle P_3 ,

114 : $P_1 \oplus 3$ bulles P_4 ,

123 : $P_1 \oplus$ bulle P_3 ,

124 : $P_3 \oplus 3$ bulles P_4 somme non directe.

der : DOUBLE ARRAY. Les valeurs de **der** = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K .

de : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

A, B : DOUBLE ARRAY. Vecteurs lignes désignant, en général, les coordonnées en x et y des points de coordonnées barycentriques L1, L2 et L3. Ces variables sont aussi notées Ac et Bc.

coeff : DOUBLE ARRAY. Vecteur ligne des coefficients de la formule de quadrature.

cas : DOUBLE ARRAY. Scalaire prenant les valeurs 1, 2, 3, 4. Choix du type de terme à calculer : duplication de **coeff** (**cas** = 1), dérivation de la fonction ω représentée par **omega** (**cas** = 2) et calcul des fonctions de base et de leurs dérivées (**cas** = 3 ou 4). La valeur **cas** = 4 donne la matrice transposée du cas où **cas** = 3.

X, Y : DOUBLE ARRAY. Vecteurs lignes ou colonnes telles que (X(1), Y(1)), (X(2), Y(2)), (X(3), Y(3)) sont les coordonnées des trois sommets du triangle courant.

L1, L2, L3 : DOUBLE ARRAY. Vecteurs lignes de tailles identiques de coordonnées barycentriques relativement au triangle de sommets de coordonnées (a1, a2), (b1, b2), (c1, c2).

Variables liées aux conditions au bord (répertoire Assemblage, trait_cb).

Nb : CELL ARRAY. Nb{ i } : est soit vide soit un vecteur colonne de DOUBLE ARRAY. Dans ce cas Nb{ i } est un vecteur qui contient les numéros de *ddl* sur lesquels on a une condition de Dirichtlet pour la variable i et ce vecteur ne contient pas d'autres termes. Le vecteur Nb{ i } peut contenir plusieurs fois le même *ddl*, par exemple lorsque celui est commun à deux frontières

Sb : CELL ARRAY. Sb{ i } : Vecteur colonne SPARSE ARRAY de taille $\dim(E_i) = \text{Dnc}(i)$, i -ème espace d'EF de l'espace EF produit E . Valeurs de la composante de la variable i au bord (Dirichtlet). La valeur de la composante au *ddl* numéro Nb{ i }(j) est donnée par Sb{ i }(Nb{ i }(j)) et Sb{ i }(k) est nul pour les autres *ddl* k qui n'appartiennent pas au vecteur Nb{ i }.

4.3 Description des fonctions et algorithmes.

Dans ce paragraphe nous décrivons les fonctions du code et, suivant la fonction, nous décrivons aussi son algorithme. Pour les algorithmes on pourra aussi se reporter au fichier contenant la fonction.

Les fonctions sont classées par répertoire avec, si possible, dans un ordre hiérarchique d'appel inverse. On commence tout d'abord par les fonctions principales du répertoire EF.

Pour les fonctions du répertoire de maillage **trait_maillage** et pour la fonction **verif_don_EF** du répertoire **utilitaires** non traitées dans ce Paragraphe, on consultera les Paragraphes 2.4 et 2.5 pour leur utilisation.

Fonctions de EF.

Fonction :

```
function path_EF ;
```

A lancer avant d'utiliser le code. Cette fonction ajoute au **path** les chemins des répertoires contenant le code. Cette fonction n'est pas appelée par les autres fonctions du code. Suivant l'environnement, on pourra soit modifier cette fonction, soit créer une fonction adaptée à l'environnement qui ajoute les chemins nécessaires, *cf.* par exemple la fonction **path_EF_unix**.

Fonction :

```
function EF_Prepa(donnees, varargin) ;
```

Cette fonction met principalement en œuvre la fonction **EF_prepa_don** pour le calcul du fichier relatif aux EF et aux formules d'intégration utilisés et qui est décrit au Paragraphe 2.2.3.

Variables d'entrée :

donnees : **STRUCT ARRAY**. Variable décrivant les données du problème EF. **varargin** : Variable pour un nombre variable d'arguments d'entrée, non utilisée dans la fonction.

Variables de sortie :**Fichiers de sortie :**

Fichier relatif aux types d'espaces d'EF utilisés et au choix des formules d'intégration de nom **donnees.nom_prep** *cf.* Paragraphe 2.2.3.

Algorithme :

```
EF/EF_prepa.m
```

On remet ici la partie déclarative de la fonction :

```
function EF_Prepa(donnees, varargin) ;
```

- Chargement de la structure du maillage et affichage d'informations sur le maillage,
 - Lancement du chronomètre,
 - Appel de la fonction **EF_prepa_don** par :

```
EF_prepa_don(donnees) ;
```
 - Affichage du temps écoulé.
-

Fonction :

```
function EF_prepa_don(donnees) ;
```

Calcul du fichier relatif aux EF et aux formules de quadrature.

Variables d'entrée :

`donnees` : STRUCT ARRAY. Variable décrivant les données du problème EF.

`varargin` : Non utilisée, nombre variable d'arguments d'entrée.

Variables de sortie :

Les variables calculées sont stockées dans un fichier. Elles sont données dans la liste du Paragraphe 4.2 :

Variables contenant des matrices d'assemblage élémentaires :

`Ass_Dom`, `Ass_Fr`, `Ass_Ad`,

Variables contenant des matrices d'assemblage globales :

`SL_Dom`, `SL_Fr`, `SL_Ad`,

Variables contenant des estimations des taille de matrices d'assemblage globales :

`Nalloc_Dom`, `Nalloc_Fr`, `Nalloc_Ad`,

Variables de ddl

`Tc`, `Xc`, `Yc`, `Arc`, `Ed`,

Variables liées à la géométrie du système :

`Pf`, `Pa`, `Pfc`.

Création de fichiers : Crée le fichier relatif aux types d'espaces d'EF utilisés et au choix des formules d'intégration de nom `donnees.nom_prep`, cf. Paragraphe 2.2.3.

Algorithme :

EF/EF_prepa_don.m

On remet ici la partie déclarative de la fonction :

```
function EF_prepa_don(donnees) ;
```

- Initialisations diverses :

```
nom_prep = donnees.nom_prep ;
```

```
nom_mail = donnees.nom_mail ;
```

```
D=donnees.D ;
```

- Chargement du maillage

```
load(nom_mail, 'Front', 'Noeu', 'T', 'X', 'Y') ;
```

- Déclaration de matrices d'assemblage du Tableau 2.2.3-1a, Paragraphe 2.2.3, et de compteurs pour les estimations des tailles de matrices EF

```
SL_Dom=cell(length(D.E),length(D.E)) ;
SL_Fr=cell(length(D.E),length(D.E)) ;
SL_Ad=cell(length(D.E),length(D.E)) ;
%Compteurs :
f_Dom=cell(length(D.E),length(D.E)) ;
f_Fr=cell(length(D.E),length(D.E)) ;
f_Ad=cell(length(D.E),length(D.E)) ;
```

- Numérotation des arêtes et description par arêtes du maillage :

Am : Tableau de deux colonnes, **Am**(*i*, :) est le vecteur des deux numéros de sommets de l'arête numéro *i* dans la numérotation globale des arêtes.

Pa : Tableau de cinq colonnes, **Pa**(*i*, :) est un vecteur de cinq termes comportant successivement le numéro du premier triangle auquel appartient l'arête *i*, son numéro local sur le triangle, puis le numéro de l'éventuel triangle et son numéro local sur le triangle ou bien les valeurs 0 et 0 pour une arête sur la frontière, ainsi que le numéro du morceau de frontière auquel appartient l'arête avec la valeur 0 pour une arête intérieure. **Pa**(:,5) est mis à 0 si **Front** = [].

Pfc : Vecteur de cellules, chaque cellule décrivant un morceau de frontière. Cette variable établit une correspondance entre les arêtes renumérotées localement sur chaque morceaux de frontière (à partir de **Front**) et leur unique triangle d'appartenance, leur numéro local d'arête sur ce triangle et leur numéro global d'arêtes. Ainsi :

Pfc{*j*}(*i*, :) est le vecteur ligne de 3 termes qui contient successivement le numéro du triangle d'appartenance de l'arête numéro *i* dans sa numérotation locale sur le morceaux de frontière *j*, son numéro local d'arête sur le triangle et son numéro global d'arête.

Pf : Tableau de 4 colonnes. établit une correspondance entre les arêtes renumérotées localement sur la frontière (à partir de l'ordre des cellules de **Pfc**) et leur unique triangle d'appartenance, leur numéro local d'arête sur ce triangle leur numéro global d'arêtes et le numéro du morceau de frontière auxquelles elles appartiennent. Ainsi :

Pf(*i*, :) est le vecteur ligne de 4 termes qui contient successivement le numéro du triangle d'appartenance de l'arête numéro *i* dans sa numérotation locale sur la frontière Γ , son numéro local d'arête sur le triangle, son numéro global d'arête et enfin **Pf**(*i*,4) qui est le numéro du morceau de frontière sur laquelle cette arête est placée.

Les variables **Pfc** et **Pf** contiennent la même information, la variable **Pfc** est utilisée pour les conditions au bord dans la fonction **cb**, l'ordre établi par **Pf** est celui utilisé pour les assemblages des termes frontière.

Les variables Pf et Pfc sont mises à 0 si Front = 0 pour des traitement autres que l'assemblage comme dans le graphisme.

```
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, Y, Front);
```

- Calcul des degrés de libertés pour chaque type d'espace d'EF composant l'espace élément fini produit E courant. Les résultats sont stockés dans les cellules des variables Tc, Xc, Yc, Arc et le vecteur de réels Ed décrits au Tableau 2.2.3-1b) du Paragraphe 2.2.3.

```
Pour i=1 :length(D.E)%boucle sur les types
    %d'espaces EF utilises
    [Td, Ad, X2, Y2] = maillage_ddl(D.E(i), T, Am, Pa, X, Y);
    Tc{i} = Td;%degres de libertes (ddl) par triangles
    Xc{i} = X2;%composantes en x des ddl
    Yc{i}= Y2;%composantes en y des ddl
    Ed(i)=length(Xc{i});%nombre de ddl de l'espace
    Arc{i}= Ad;%description des ddl d'aretes
Fin;
```

- Suite de la déclaration des matrices Tableau 2.2.3-1a, Paragraphe 2.2.3 :

```
Pour d1=1 :length(D.der);%derivations
    Pour d2=1 :length(D.der);%derivations
        Pour i=1 :length(D.E)
            Pour j=1 :length(D.E)
                SL_Dom{i,j,d1,d2}=sparse(Ed(i),Ed(j));
                SL_Fr{i,j,d1,d2}=sparse(Ed(i),Ed(j));
                %aretes interieures et exterieures =
                % aretes du domaine :
                SL_Ad{i,j,d1,d2}=sparse(Ed(i),Ed(j));
            Fin;
        Fin;
    Fin;
Fin;
%Compteurs vus ci-dessus :
Pour i=1 :length(D.E)
    Pour j=1 :length(D.E)
        f_Dom{i,j}=sparse(Ed(i),Ed(j));
        f_Fr{i,j}=sparse(Ed(i),Ed(j));
```



```

        f_Ad{i,j}=sparse(Ed(i),Ed(j)) ;
    Fin ;
Fin ;
• Boucles pour l'assemblage.
- Boucle sur les triangles :
    s=size(T) ;
    Ass_Dom=[] ;
    Pour i=1 :s(1)
        [Ass_Dom, SL_Dom, f_Dom] = ...
            matrice_elem_prepa_T(donnees, i, T,...
                Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;
    Fin
- Boucle sur les arêtes frontières :
    Ass_Fr=[] ;
    s=size(Pf) ;
    %on decrit l'ensemble des aretes frontieres avec leur
    %  numerotation locale sur la frontiere
    Pour i=1 :s(1)
        [Ass_Fr, SL_Fr, f_Fr] = ...
            matrice_elem_prepa_A(donnees, i, T, ...
                Tc, Pf, X, Y, D, Ass_Fr, SL_Fr, f_Fr) ;
    Fin ;
- Boucle sur les arêtes intérieures et les arêtes frontières (actuellement utilisé pour
Lesaint-Raviart). On tient compte de chaque côté de l'arête pour tenir compte
dérivations sur les arêtes :
    Ass_Ad=[] ;
    s=size(Pa) ;
    %on decrit l'ensemble des aretes avec leur numerotation globale
    Pour i=1 :s(1)
        [Ass_Ad, SL_Ad, f_Ad] = ...
            matrice_elem_prepa_Ad(donnees, i, T, ...
                Tc, Pa, X, Y, D, Ass_Ad, SL_Ad, f_Ad) ;
    Fin ;
• Estimation du nombre de terme non nul des matrices : (cf. Tableau 2.2.3-1a,
Paragraphe 2.2.3) :

```

```

Pour i=1 :length(D.E)
    Pour j=1 :length(D.E)
        Nalloc_Dom(i,j) = full(sum(sum(f_Dom{i,j})));
        Nalloc_Fr(i,j) = full(sum(sum(f_Fr{i,j})));
        Nalloc_Ad(i,j) = full(sum(sum(f_Ad{i,j})));
    Fin;
Fin;

```

- Stockage des données des Tableaux 2.2.3-1a) et b) du Paragraphe 2.2.3.

```

save(nom_prep, 'Ass_Dom', 'Ass_Fr', 'Ass_Ad', ...
      'SL_Dom', 'SL_Fr', 'SL_Ad', ...
      'Nalloc_Dom', 'Nalloc_Fr', 'Nalloc_Ad', ...
      'Tc', 'Pf', 'Pa', 'Arc', 'Pfc', 'Xc', 'Yc', 'Ed' );

```

- Fin de la fonction

Exemple : L'exemple suivant calcule les variables calculées et sauvées par EF_prepa_don ci-dessus dans le cas du maillage de 8 triangles de la Figure 2.3.2-1 du Paragraphe 2.3.2 et des éléments P_1^c et P_2^c dont les *ddl* sont donnés à la figure 2.3.2-2c. La commande pour cet exemple est

```
cd C :/EF/exemples, exemple_EF_prepa_don
```

et le fichier de commandes est donné par :

Fichier C :/EF/exemples/exemple_EF_prepa_don.m

```

%Utilisation de EF/EF_prepa_don.
%  Calculs relatifs aux espaces EF.
%  Commande :
%      cd C :/EF/exemples, exemple_EF_prepa_don

clear
close all
%Structure du maillage (8 triangles, utilises comme
%exemples pour illustrer la numerotation des DDL globaux)
T = [1,5,4; 1,2,5; 2,6,5; 2,3,6; 4,8,7; 4,5,8; 5,9,8; 5,6,9];
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1];
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1];
Noeu = 9;
Front{1} = [7,8,9];
Front{2,1} = [9,6,3]; %mise en colonne
Front{3} = [3,2,1];

```

```
Front{4} = [1,4,7] ;

%variable decrivant les espaces d'EF
%Types d'espaces d'EF : E1 = P1 et E2 = P2
D.E = [1,2] ;

%Derivation (toujours fixe dans ce code)
D.der = [0,1,2] ;

%Numero de la formule de quadrature sur les triangles
D.intT = 4 ;

%Numero de la formule de quadrature sur les aretes
D.intA = 4 ;

%Choix de l'espace produit : ici E = E1 x E2 x E1
D.f = [1,2,1] ;

%Dimensions locales des espaces EF composant l'espace produit
D.d = [3,6,3] ;

%Structure de donnees decrivant le probleme EF
donnees.D = D ;

%Nom du fichier de maillage
donnees.nom_mail = ...
    'C :/EF/exemples/rep_exemples_mat/exemple_EF_prepa_don_mail' ;

%Sauvegarde des donnees de maillages
save(donnees.nom_mail,'X','Y','T','Noeu','Front')

%Nom du fichier de matrices
donnees.nom_prep = ...
    'C :/EF/exemples/rep_exemples_mat/exemple_EF_prepa_don_mat' ;

%Utilisation de EF_prepa_don
EF_prepa_don(donnees) ;

%Chargement des donnees calculees par EF_prepa_don
```

```

load(donnees.nom_prep,'Ass_Dom', 'Ass_Fr', 'Ass_Ad', ...
      'SL_Dom', 'SL_Fr', 'SL_Ad', ...
      'Nalloc_Dom', 'Nalloc_Fr', 'Nalloc_Ad', ...
      'Tc' , 'Pf', 'Pa', 'Arc', 'Pfc', 'Xc', 'Yc', 'Ed' );

%Trace du maillage
for i = 1 :size(T)
    hold on
    patch(X(T(i, :)), Y(T(i, :)), 'w')
end;

for i = 1 :length(X)
    text(X(i) + 0.02, Y(i) + 0.02,num2str(i))
end;

%Exemple trace des ddl d'aretes de donnees par Arc
plot(Xc{1}(Arc{1}), Yc{1}(Arc{1}), 'o');
plot(Xc{2}(Arc{2}), Yc{2}(Arc{2}), 'bx');

```

Fonction :

```
function [S,F] = EF_assemblage(donnees, Uc) ;
```

Assemblage du système.

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF.

Uc : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (cf. Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

Variables de sortie :

S : DOUBLE ARRAY. Matrice finale du système, après assemblage sur l'espace EF produit courant E , après changement de base par rotations successives pour les conditions au bord portant sur les composantes normales ou tangentielles de champs de vecteurs, changement de base pour des conditions de type $AX = b$ sur le bord, transformation du système obtenu pour la prise en compte des conditions au bord (cf. (1.4)) et retour à la base initiale et enfin élimination d'inconnues du système : inconnues composantes (cf. (1.3)') de l'espace EF produit courant ou inconnues ponctuelles.

f : DOUBLE ARRAY. Représente le second membre final, tenant compte des conditions au bord et l'élimination d'inconnues.

Algorithme :

EF/EF_assemblage.m

On remet ici la partie déclarative de la fonction :

```
function [S,F] = EF_assemblage(donnees, Uc) ;
```

- Initialisations diverses dont D variable décrivant les espaces d'EF (Paragraphe 2.2.1 et Tableau 2.2.1-1c et d) :

```
D = donnees.D ;
nom_prep = donnees.nom_prep ;
nom_mail = donnees.nom_mail ;
```

- Chargement du maillage (Paragraphe 2.2.2 et Tableau 2.2.2-1) :

```
load(nom_mail, 'Front', 'Noeu', 'T', 'X', 'Y') ;
```

- Chargement de données du fichier relatif aux EF et aux *ddl* (Paragraphe 2.2.1, Tableau 2.2.1-1a), les données chargées ici étant décrites au Paragraphe 2.2.3, Tableau 2.2.3-1b :

```
load(nom_prep, 'Arc', 'Pfc', 'Xc', 'Yc', 'Ed') ;
```

- Nombre de degrés de liberté par espace :

```
Dnc = Ed(D.f) ;
```

- Prise en compte des conditions au bord de type Dirichtlet, le champ `donnees.fcb` est décrit au Paragraphe 2.2.1, Tableau 2.2.1-1b, il désigne la fonction décrivant les conditions au bord (Paragraphe 2.2.4, Tableau 2.2.4-2). Ci-dessous $Nb\{i\}$ représente les *ddl* sur lesquelles on a une condition de type Dirichtlet sur la variable i et $Sb\{i\}$ la valeur de la variable i en ce *ddl*, avec une variante dans le cas de conditions normale ou tangentes comme expliqué au 2.2.4, Tableau 2.2.4-2 :

```
[Nb,Sb] = cb(donnees,donnees.fcb,T,Arc,Pfc,Xc,Yc,Dnc,Front) ;
```

- Initialisation des matrices et des seconds membre de type CELL :
- Matrice du système de l'assemblage sur les triangles :

```
m1 = length(D.f) ;
Sc_Dom = cell(m1,m1) ;
Fc_Dom_scm = cell(m1,1) ;
```

- Matrice du système de l'assemblage sur les frontières :

```
Sc_Fr = cell(m1,m1) ;
Fc_Fr_scm = cell(m1,1) ;
```

- Matrice du système de l'assemblage sur les arêtes du maillage pour la prise en compte de la méthode de Lesaint-Raviart et, en développement, de la forme bilinéaire a_{Σ} de (1.5) et de la forme linéaire b_{Σ} de (1.10) :

```
Sc_Ad = cell(m1,m1) ;
Fc_Ad_scm = cell(m1,1) ;
```

- Chargement des valeurs estimant la taille des matrices du système (Paragraphe 2.2.3, Tableau 2.2.3-1a) puis allocation mémoire des cellules des matrices du système ci-dessus. Chaque cellule est une matrice de type **SPARSE ARRAY** :

```
load(donnees.nom_prep, 'Nalloc_Dom', 'Nalloc_Fr', 'Nalloc_Ad') ;
```

- Boucles pour l'assemblage. Utilisation dans ces boucles, des fonctions désignées par les champs de **donnees** du Paragraphe 2.2.1, Tableau 2.2.1-1b et du Tableau 2.2.1-1d, pour la variable **D.LR** qui définit les fonctions champs de vecteurs suivant lesquels on dérive.

- Chargement des matrices élémentaires pré-calculées et du tableau des *ddl* (Paragraphe 2.2.1, Tableau 2.2.1-1a pour le champ **donnees.nom_prep**, Paragraphe 2.2.3, Tableau 2.2.3-1a pour le contenu de **Ass_Dom** et Paragraphe 2.2.3, Tableau 2.2.3-1b pour les données des *ddl*) :

```
load(donnees.nom_prep, 'Ass_Dom', 'Tc') ;
```

- Boucle sur les triangles pour l'assemblage sur le domaine. Second membre :

```
Mat_syst_fscm_Dom = donnees.fscm_Dom ;
Pour i=1 :size(T,1)
    [Fc_Dom_scm] = matrice_elem_T_scm(donnees, i, T, Tc, ...
        X, Y, Uc, D, Mat_syst_fscm_Dom, Ass_Dom, Fc_Dom_scm) ;
Fin ;
```

- Boucle sur les triangles pour l'assemblage sur le domaine. Membre de gauche :

```
Mat_syst_Dom = donnees.feq_Dom
Pour i=1 :size(T,1)
    [Sc_Dom] = matrice_elem_T(donnees, i, T, Tc, ...
        X, Y, Uc, D, Mat_syst_Dom, Ass_Dom, Sc_Dom) ;
Fin ;
```

- Chargement des matrices élémentaires pré-calculées et du pointeur d'arêtes frontières (Paragraphe 2.2.3, Tableau 2.2.3-1a et b) :

```
load(nom_prep, 'Ass_Fr', 'Pf') ;
```

- Boucle sur les arêtes de la frontière pour l'assemblage sur la frontière. Second membre :

```
Mat_syst_fscm_Fr=donnees.fscm_Fr ;
```

```

Pour i=1 :size(Pf,1)
    [Fc_Fr_scm] = matrice_elem_A_scm(donnees, i, T, Tc, ...
        Pf, X, Y, Uc, D, Mat_syst_fscm_Fr, Ass_Fr, Fc_Fr_scm);
Fin;

```

- Boucle sur les arêtes de la frontière pour l'assemblage sur la frontière. Membre de gauche :

```

Mat_syst_Fr = donnees.feq_Fr;
Pour i=1 :size(Pf,1)
    [Sc_Fr] = matrice_elem_A(donnees, i, T, Tc, ...
        Pf, X, Y, Uc, D, Mat_syst_Fr, Ass_Fr, Sc_Fr);
Fin;

```

- Chargement des matrices élémentaires pré-calculées et du pointeur d'arêtes du maillage (Paragraphe 2.2.3, Tableau 2.2.3-1a et b) :

```
load(nom_prep, 'Ass_Ad', 'Pa');
```

- Boucle sur les arêtes du maillage pour l'assemblage de b_{Σ} . Second membre :

```

Mat_syst_fscm_Ad = donnees.fscm_Ad;
Pour i=1 :size(Pa,1)
    [Fc_Ad_scm] = matrice_elem_Ad_scm(donnees, i, T, Tc, ...
        Pa, X, Y, Uc, D, Mat_syst_fscm_Ad, Ass_Ad, Fc_Ad_scm);
Fin;

```

- Boucle sur les arêtes du maillage pour l'assemblage de a_{Σ} . Membre de gauche :

```

Mat_syst_Ad = donnees.feq_Ad;
Pour i=1 :size(Pa,1)
    [Sc_Ad] = matrice_elem_Ad(donnees, i, T, Tc, ...
        Pa, X, Y, Uc, D, Mat_syst_Ad, Ass_Ad, Sc_Ad);
Fin;

```

- Boucle sur les arêtes du maillage pour la méthode de Lesaint-Raviart. Les inconnues sur lesquelles intervient la dérivation sont données par la deuxième colonne du tableau D.LR qui est de type CELL, la première colonne contenant des handles sur les fonctions définissant les champs de vecteurs suivant lesquels on dérive (Paragraphe 2.2.1, Tableau 2.2.1-1d) :

```

Pour i=1 :size(Pa,1);
    Pour iD=1 :size(D.LR,1);
        [Sc_Ad] = matrice_elem_Ad_int_LR(donnees, i, T, Tc, ...

```

```

Pa, X, Y, Uc, D, D.LR{iD,1}, D.LR{iD,2}, ...
Ass_Ad, Sc_Ad) ;

Fin ;

Fin ;

```

Dans cette boucle seules les contributions des arêtes intérieures sont prises en compte.

- Addition des trois contributions à la matrice et au second membre du système avant conditions au bord et éliminations de certains *ddl*. Utilisation de `ADD_MAT_CELL` qui additionne des tableaux de cellules. La matrice du système avant conditions au bord et élimination d'inconnues est donnée par `Sc` et le second membre par `Fc` :

```

Sc = add_mat_cell(Sc_Dom, Sc_Fr) ;
Sc = add_mat_cell(Sc, Sc_Ad) ;
Fc = add_mat_cell(Fc_Dom_scm , Fc_Fr_scm) ;
Fc = add_mat_cell(Fc,Fc_Ad_scm) ;

```

Les matrices calculées sont celles du système obtenu en remplaçant $\mathcal{E}$ par E dans le système (1.3) :

$$\exists \tilde{X} \in E, \mathcal{A}(X^0, X, Y) = \mathcal{B}(X^0, F, Y), \forall Y \in E.$$

- Initialisation des variables du 2.2.4, Tableau 2.2.4-2, décrivant les conditions au bord :

```
[fb,sb] = feval(donnees.fcb,donnees) ;
```

- Prise en compte de conditions de type Dirichlet portant sur des composantes normales et tangentes de couples d'inconnues. Si des contraintes de ce type sont imposées, les matrices du système et le second membre subissent une transformation type changement de base $A = P'AP$ et $f = P'F$, avec P matrice de passage orthogonale :

- Calcul de la matrice de changement de base P . Boucle sur le nombre de couples sur lesquels il y a un condition de ce type (Paragraphe 2.2.4, Tableau 2.2.4-2) :

```

P = speye(sum(Dnc)) ;
Pour i=1 :size(sb.un,1) ;

```

Cas des normales sur les coins pondérées (cf Paragraphe 3.2.2) :

```

Si length(sb.un(i, :)) = 2
    P = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
        donnees.D, sb.un{i,1}(1), sb.un{i,1}(2), ...
        sb.un{i,2}) ;

```


Fin ;

Cas d'une condition sur les composantes (n', t') du couple choisi, avec (n', t') vecteurs obtenu par rotation de la normale n et du vecteur tangent t avec une rotation d'angle donnée par `sb.un{i,3}` avec la valeur d'angle 0 si `sb.un{i,3} = []` (cf. Tableau 2.2.4-2) :

```
Si length(sb.un(i, :)) = 3
%Rotation eventuelle
Pour i=1 :size(sb.un,1) ;
    P = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
        donnees.D, sb.un{i,1}(1), sb.un{i,1}(2), ...
        sb.un{i,2}, [], sb.un{i,3}) ;
```

Fin ;

Traitement d'une éventuelle rotation et d'un ordre de priorité de normales situées à l'intersection de morceaux de frontière, cet ordre est indiqué par la matrice `sb.un{i,4}` (cf. Tableau 2.2.4-2) :

```
Si length(sb.un(i, :)) = 4
%Cas des normales sur les coins ponderees
P = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
    donnees.D, sb.un{i,1}(1), sb.un{i,1}(2), ...
    sb.un{i,2}, sb.un{i,4}, sb.un{i,3}) ;
```

Fin ;

Fin ;

Ci dessus P est une matrice orthogonale telle que $P'P = Id$.

- Système dans la nouvelle base. On transforme les matrices de type `CELL` avec `TRANS_MAT_CELL_MAT` en matrices du type `DOUBLE`. Ensuite on effectue le changement de base, puis on revient à des matrices de type `CELL` avec `TRANS_MAT_MAT_CELL` :

```
S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;
S = P'*S*P ;
F = P'*F ;
Sc = trans_mat_mat_cell(S,D,Dnc) ;
Fc = trans_mat_mat_cell(F,D,Dnc) ;
```

• Prise en compte de conditions de type Dirichlet de type $AX = U_0$ (cf. 2.2.4, Tableau 2.2.4-2). Si des contraintes de ce type sont imposées, les matrices du système et le second membre subissent une transformation type changement de

base $S = P'SP$ et $f = P'F$, avec P matrice de passage déduite de A^{-1} ou bien de B^{-1} où B est une matrice extraite de A (cf. 2.2.4, Tableau 2.2.4-2, et le commentaire qui suit ce tableau). Ci-dessous la valeur de P est prise par PA :

- Calcul de la matrice de changement de base P . Boucle sur le nombre de couples sur lesquels il y a une condition de ce type (Paragraphe 2.2.4, Tableau 2.2.4-2) :

```
P1=speye(sum(Dnc)) ;
PA=speye(sum(Dnc)) ;
PAinv=speye(sum(Dnc)) ;
Pour i=1 :size(sb.a,1) ;
    %valeur de l'inverse de A :
    AI=inv(sb.a{i,3}) ;
```

sb.a a trois ou 4 colonnes, la quatrième colonne permet de modifier le choix du changement de base à l'intersection de deux frontières :

```
Si size(sb.a,2)==4 ;
    [PA, PAinv] = changt_base_a(PA, PAinv, T, Xc, Yc, ...
        Arc, Pfc, Dnc, D, sb.a{i,1}, sb.a{i,2}, AI, ...
        Front, sb.a{i,4}) ;
sinon% sb.a a ici necessairement 3 colonnes
    [PA, PAinv] = changt_base_a(PA, PAinv, T, Xc, ...
        Yc, Arc, Pfc, Dnc, D, sb.a{i,1}, sb.a{i,2}, AI) ;
Fin ;
Fin ;
```

- Système dans la nouvelle base. On transforme les matrices de type CELL avec TRANS_MAT_CELL_MAT en matrices du type DOUBLE. Ensuite on effectue le changement de base, puis on revient à des matrices de type CELL avec TRANS_MAT_MAT_CELL :

```
S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;
S = PA'*S*PA ;
F = PA'*F ;
Sc = trans_mat_mat_cell(S,D,Dnc) ;
Fc = trans_mat_mat_cell(F,D,Dnc) ;
```

• Transformation du système pour la prise en compte des conditions au bord, dans la nouvelle base, cette transformation est effectuée si Nb est non vide :

```
[Sc,Fc] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;
```

- Cas des conditions de type $AX = U_0$: retour à la base de départ. Les opérations sont à nouveau réalisées sur les matrices de type DOUBLE :

```

S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;
S = PAinv'*S*PAinv ;
F = PAinv'*F ;
Sc = trans_mat_mat_cell(S,D,Dnc) ;
Fc = trans_mat_mat_cell(F,D,Dnc) ;

```

- Cas des conditions sur les normales : retour à la base de départ. Les opérations sont à nouveau réalisées sur les matrices de type DOUBLE :

```

S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;
S = P*S*P' ;
F = P*F ;
Sc = trans_mat_mat_cell(S,D,Dnc) ;
Fc = trans_mat_mat_cell(F,D,Dnc) ;

```

- Elimination d'inconnues ponctuelles indiquées par la variable `D.inc_point` :

```

Sc = supp_lin_col_mat_cell(Sc,D) ;
Fc = supp_lin_col_vec_cell(Fc,D) ;

```

- Elimination d'inconnues par cellules :

Pour `i= D.inc_cell`

```

Sc( :,i(1)) = add_mat_cell(Sc( :,i(1)),Sc( :,i(2))) ;
j = D.inc_cell(2, :) ;
Sc(j, :) = [] ;
Sc( :,j) = [] ;
Fc(j) = [] ;

```

fin

- Effacement des variables à l'exception de `Sc` et `Fc`.
- Mise du système sous forme de matrice de type DOUBLE :

```

S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;

```

Exemples : Donnons quatre exemples d'utilisation de la fonction `EF_assemblage` appliquée à la résolution d'un Laplacien, d'un écoulement de Poiseuille, d'un point fixe appliqué à la loi puissance et du problème de la cavité entraînée. La résolution de ces problèmes est effectuée sur le carré unité $\Omega =]0, 1[^2$ muni du maillage suivant :

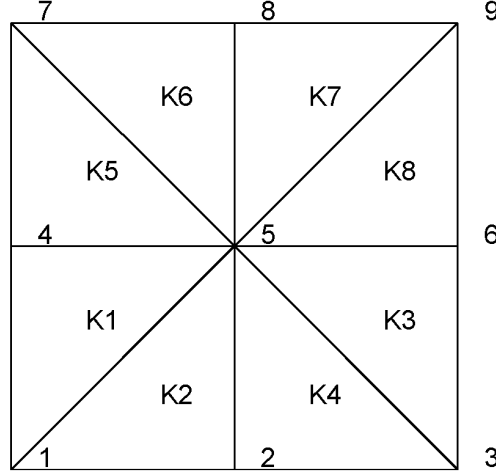


Figure 4.3-1. Maillage par 8 triangles du carré unité.

La frontière Γ de Ω est partitionnée en 4 segments $\Gamma_1 = [0, 1] \times \{1\}$, $\Gamma_2 = \{1\} \times [0, 1]$, $\Gamma_3 = [0, 1] \times \{0\}$ et $\Gamma_4 = \{0\} \times [0, 1]$.

Exemple 1.

Le fichier de fonction `EF/exemples/exemple1-EF_assemblage` est consacré à l'approximation du problème du Laplacien :

$$\exists u \text{ tel que } \begin{cases} -\Delta u = -6x - 6y \text{ sur } \Omega \\ u = x^3 + y^3 \text{ sur } \Gamma, \end{cases}$$

de solution

$$u(x, y) = x^3 + y^3.$$

La formulation variationnelle classique de ce problème est la suivante :

$$\exists u \in H^1(\Omega) \text{ tel que : } \int_{\Omega} u_{,x} v_{,x} + u_{,y} v_{,y} = \int_{\Omega} f v, \quad \forall v \in H_0^1(\Omega), \quad u = g \text{ sur } \Gamma,$$

avec $f(x, y) = -6x - 6y$ et $g(x, y) = x^3 + y^3$ ou encore

$$\begin{aligned} \exists u \in H^1(\Omega) \text{ tel que : } \int_{\Omega} \left(\sum_{0 \leq i, j \leq 2} (M_{ij} u_{,i}, N_{ij} v_{,j}) \right) = \\ \int_{\Omega} \sum_{0 \leq i, j \leq 2} (\tilde{M}_{ij} f_i, \tilde{N}_{ij} v_{,j}), \quad \forall v \in H_0^1(\Omega), \quad u = g \text{ sur } \Gamma, \end{aligned}$$

avec

$$M_{ij} = N_{ij} = 0, 0 \leq i, j \leq 2, \text{ pour } (i, j) \notin \{(1, 1), (2, 2)\}$$

et

$$M_{11} = M_{22} = N_{11} = N_{22} = 1,$$

et pour le second membre

$$\tilde{M}_{ij} = \tilde{N}_{ij} = 0, 0 \leq i, j \leq 2, \text{ pour } (i, j) \neq (0, 0) \text{ et } \tilde{M}_{00} = \tilde{N}_{00} = 1$$

et

$$f_1 = -6x - 6y, f_2 = f_3 = 0.$$

Dans le fichier `exemple`, la forme bilinéaire est donnée par la sous-fonction `eq_Dom`, le second membre par la sous-fonction `scm_Dom` et les conditions aux bord par la sous-fonction `cb_don`. Dans ce fichier, pour avoir une vue d'ensemble, on a regroupé toute la structure de donnée. En général cette structure sera rangée dans divers fichiers et répertoires : maillage, fichiers de fonctions, fonction d'initialisation de la variable `donnees` au lieu d'une initialisation directe dans le fichier d'instructions comme ici.

Les fonctions `eq_Dom` et `scm_Dom` renvoient les variables suivantes. Soit K le triangle courant de numéro `numK`, notons $m_K = (x_1, y_1), \dots, (x_\alpha, y_\alpha)$ les coordonnées cartésiennes des points de la formule de quadrature et posons $\mathbf{Zero} = (0, \dots, 0)$ et $\mathbf{Un} = (1, \dots, 1)$ les vecteurs de longueur α , le nombre de points de la formule de quadrature.

- La fonction `eq_Dom` renvoie alors des matrices de type `CELL ARRAY` (cellules) `Mc`, `Nc` de taille 3×3 qui correspond aux 3 indices de dérivations. Les termes $M\{i, j\}$ et $N\{i, j\}$ sont des matrices de type `CELL ARRAY` dont les tailles dépendent de la forme bilinéaire attachée au problème et qui ici sont de taille 1×1 . On a $Mc\{1, 1\} = \{\mathbf{Zero}\}$ ou de manière équivalente $Mc\{1, 1\}\{1, 1\} = \mathbf{Zero}$, qui correspond à la valeur de $M_{00} = 0$, $Mc\{2, 2\} = \{\mathbf{Un}\}$ pour $M_{11} = 1$, $Mc\{1, 2\} = \{\mathbf{Zero}\}$ pour $M_{01} = 0$ etc.

- La fonction `scm_Dom` renvoie des matrices `Mc` et `Nc` du même type que celles renvoyées par `eq_Dom`, ces matrices correspondent aux valeurs des matrices $\tilde{M}_{ij}$ et $\tilde{N}_{ij}$, $0 \leq i, j \leq 2$ ci-dessus. La fonction `scm_Dom` renvoie aussi la vecteur `Fc` de taille 1×3 correspondant aux valeurs du second membre f_i , avec $i = 1, 2, 3$ pour les trois dérivations. Ici $Fc\{i\}$ est un vecteur ligne, ici de taille 1×1 , avec $Fc\{1\} = \{f(m_K)\}$ et $Fc\{2\} = Fc\{3\} = \{\mathbf{Zero}\}$.

La fonction `cb_don` renvoie dans le vecteur ligne `fb` de type `CELL ARRAY` qui décrit les numéros de frontières sur lesquelles on a une condition au bord. Ici `fb` est de taille 1×1 de cellule `fb\{1\} = [1, 2, 3, 4]`. Ensuite `cb_don` renvoie aussi une deuxième variable `sb` qui est de type `struct` dont le champ `sb.v` est un tableau 1×4 de type `CELL`, dont la cellule $(1, i)$ contient un handle sur la fonction qui donnent la condition au bord sur la frontière `fb\{1\}(i)`. On a ici `sb.v\{1, j\} = @f` pour $j = 1, \dots, 4$, `f` étant la fonction des conditions au bord du problème telle que $f(x, y) = x^3 + y^3$.

Pour ce problème on a utilisé l'approximation par du P_3 continu, espace n° 3 dans le code, qui donne la solution exacte. Le tracé de la solution trouvée et une comparaison avec la solution exacte est donnée à la fin du fichier de fonction exemple donné ci-après.

Fichier de fonction de l'exemple 1 :

EF/exemples/exemple1_EF_assemblage.m

```
% Fichier EF/exemples/exemple1_EF_assemblage.m
% -Laplacien u = -6x - 6y sur le carre unite,
% u = x.^3 + y.^3 au bord, de solution x^3 + y^3.
%
% Utilisation de EF/EF_assemblage.
% Calcul relatif aux espaces EF.
% Commande
%      cd C :/EF, path_EF, cd C :/EF/exemples
%      [Uc, donnees] = exemple1_EF_assemblage ;

function [Uc, donnees] = exemple1_EF_assemblage
close all ;
%Maillage de 8 triangles, ici chaque triangle a au plus
%  une arete sur la frontiere
T = [1,5,4 ; 1,2,5 ; 3,6,5 ; 2,3,5 ; 4,5,7 ; 7,5,8 ; 5,9,8 ; 5,6,9] ;
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = 9 ; %variable indicative dans EF_prepa_don
Front{1} = [7,8,9] ;
Front{2,1} = [9,6,3] ; %mise en colonne de Front
Front{3} = [3,2,1] ;
Front{4} = [1,4,7] ;

%Variable decrivant les espaces d'EF
%Types d'espaces d'EF : E1 = P3
D.E = [3] ;
%Derivation (toujours fixe dans ce code)
```

4.3. DESCRIPTION DES FONCTIONS FONCTION EF/EF_assemblage (*Exemple 1*)

```

D.der = [0,1,2] ;
%Numero de la formule de quadrature sur les triangles (ordre 4)
D.intT = 4 ;
%Numero de la formule de quadrature sur les aretes (ordre 4)
D.intA = 4 ;
%Choix de l'espace produit : ici E = E1
D.f = [1] ;
%Structure de donnees decrivant le probleme EF
donnees.D = D ;

%Nom du fichier de maillage
donnees.nom_mail = ...
    'C :/EF/exemples/rep_exemples_mat/exemple1_EF_assemblage_mail' ;

%Sauvegarde des donnees de maillages
save(donnees.nom_mail,'X','Y','T','Noeu','Front') ;

%Nom du fichier de matrices
donnees.nom_prep = ...
    'C :/EF/exemples/rep_exemples_mat/exemple1_EF_assemblage_mat' ;

%Conditions aux limites
donnees.fcb = @cb_don ;

%Equations (formes bilineaires) :
donnees.feq_Dom = @eq_Dom ; %sur le domaine

%Seconds membres (formes lineaires)
donnees.fscm_Dom = @scm_Dom ; %sur le domaine

%Calculs des donnees relatives aux espaces et aux formules
%   d'integration (on peut mettre aussi EF_prepa_don)
EF_prepa(donnees) ;

```

```
%Initialisation de Uc au vecteur nul de l'espace EF produit
```

```
Uc = init_transf_cell('init', [], donnees);
```

```
%Assemblage
```

```
[S,f] = EF_assemblage(donnees,Uc);
```

```
%Resolution
```

```
U = S\f;
```

```
%On remet U sous forme de cellules Uc pour le trace
```

```
U = full(U);
```

```
Uc = init_transf_cell('ta2c',U,donnees);
```

```
%Trace de la solution
```

```
trac_solEc(donnees, Uc, 0, 1, 4);
```

```
%Verification
```

```
disp('Erreur en P3 : (1.1102e-015) :');
```

```
load(donnees.nom_prep,'Xc','Yc');
```

```
erreur = max(abs(Uc{1}' - (Xc{1}.^3 + Yc{1}.^3)));
```

```
disp(erreur);
```

```
%Sous-fonctions :
```

```
%Membre de gauche
```

```
function [Mc, Nc] = eq_Dom(donnees, uc, ...
```

```
    a1, b1, c1, a2, b2, c2, de, Ac, Bc, numK);
```

```
%Membre de gauche : Laplacien.
```

```
%Termes intermediaires
```

```
Zero = zeros(1,length(Ac));
```

```
Un = ones(1,length(Ac));
```

```
%Matrices (forme non factorisee)
```

```
Mc{1,1} = {Zero}; Mc{1,2} = {Zero}; Mc{1,3} = {Zero};
```


4.3. DESCRIPTION DES FONCTIONS FONCTION EF/EF_assemblage (*Exemple 1*)

```

Mc{2,1} = {Zero} ; Mc{2,2} = {Un} ;    Mc{2,3} = {Zero} ;
Mc{3,1} = {Zero} ; Mc{3,2} = {Zero} ; Mc{3,3} = {Un} ;
Nc = Mc ;

%Second membre
function [Mc, Nc, Fc] = scm_Dom (donnees, uc, a1, b1, c1, ...
                                a2, b2, c2, de, Ac, Bc, numK) ;
%Second membre :  $f(x,y) = -6x - 6y$ .

%Termes intermediaires
Zero = zeros(1,length(Ac)) ;
Un   = ones(1,length(Ac)) ;

%Matrices
Mc{1,1} = {Un} ; Mc{1,2} = {Zero} ; Mc{1,3} = {Zero} ;
Mc{2,1} = {Zero} ; Mc{2,2} = {Zero} ; Mc{2,3} = {Zero} ;
Mc{3,1} = {Zero} ; Mc{3,2} = {Zero} ; Mc{3,3} = {Zero} ;
Nc = Mc ;

%Second membre f
Fc{1,1} = {-6*Ac - 6*Bc} ; Fc{1,2} = {Zero} ; Fc{1,3} = {Zero} ;

%Conditions au bord
function [fb,sb] = cb_don(donnees) ;
%Conditions au bord : condition de Dirichtlet  $u = x^3 + y^3$ .

%La frontiere est partitionnee en 4 arcs frontieres. On indique
% les frontieres avec Dirichtlet pour les inconnues il y a ici
% une unique inconnue de numero 1
fb{1} = [1 :4] ;
%Valeur par fonction de l'inconnue numero 1 sur les frontieres
% numerotees 1,2,3,4 ci-dessus
for i = 1 :4
    sb.v{1,i} = @f ;

```

end ;

```
%Sous-fonctions valeur des conditions au bord
function z = f(x,y)
z = x.^3 + y.^ 3 ;
```

Exemple 2.

On considère le problème de Stokes suivant posé sur le carré unité :

$$\begin{cases} \exists(u, p) \text{ tel que :} \\ -\eta\Delta u + \nabla p = f \text{ sur } \Omega \\ \nabla \cdot u = 0 \text{ sur } \Omega \\ u = u_0 \text{ sur } \Gamma, \end{cases}$$

où la vitesse u est une fonction à valeur dans $\mathbb{R}^2$ et p est la pression. Le problème est résolu avec $f = (0, 0)$ et comme condition au bord u_0 , l'écoulement de Poiseuille dans la direction de l'axe des x donné par :

$$u_0(x, 0) = u_0(x, 1) = (0, 0), \quad u_0(0, y) = u_0(1, y) = (4y(1 - y), 0)$$

La solution de ce problème est alors donnée par :

$$u(x, y) = (4y(1 - y), 0), \quad p(x, y) = -8x + C,$$

p étant défini à la constante C près.

Le problème admet la formulation variationnelle suivante :

$$\begin{cases} \exists(u, p) \in [H^1(\Omega)]^2 \times L_0^2(\Omega) \text{ tel que} \\ \int_{\Omega} 2\eta(d(u), d(v)) - (p, \nabla \cdot v) = \int_{\Omega} f v, \forall v \in [H_0^1(\Omega)]^2, \\ \int_{\Omega} (\nabla \cdot u, q) = 0, \forall q \in L_0^2(\Omega), \\ u = u_0 \text{ sur } \Gamma, \end{cases}$$

où $d(u)$ est le tenseur des vitesses de déformation $(d(u))_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i})$ et $(d(u), d(v))$ le produit scalaire :

$$\begin{aligned} (d(u), d(v)) &= \sum_{1 \leq i, j \leq 2} (d(u))_{ij} (d(v))_{ij} \\ &= u_{1,1}v_{1,1} + \frac{1}{2}(u_{1,2}v_{1,2} + u_{1,2}v_{2,1} + u_{2,1}v_{1,2} + u_{2,1}v_{2,1}) + u_{2,2}v_{2,2}. \end{aligned}$$

Le problème peut se mettre sous la forme variationnelle suivante (forme factorisée) :

$$\exists (u, p) \in [H^1(\Omega)] \times L_0^2(\Omega) \text{ tel que : } \int_{\Omega} \left(\sum_{i=0}^2 M_i \begin{pmatrix} u \\ p \end{pmatrix}_i, \sum_{i=0}^2 N_i \begin{pmatrix} v \\ q \end{pmatrix}_i \right) =$$

$$\int_{\Omega} \left(\sum_{i=0}^2 \tilde{M}_i F_i, \sum_{i=0}^2 \tilde{N}_i \begin{pmatrix} v \\ q \end{pmatrix}_i \right), \forall (v, q) \in [H^1(\Omega)] \times L_0^2(\Omega), u = u_0 \text{ sur } \Gamma,$$

avec

$$M_0 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}; M_1 = 2\eta \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ -\frac{1}{2\eta} & 0 & 0 \end{pmatrix}; M_2 = 2\eta \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2\eta} & 0 \end{pmatrix};$$

et

$$N_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}; N_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

et le second membre :

$$\tilde{M}_0 = \tilde{N}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \tilde{M}_1 = \tilde{M}_2 = \tilde{N}_1 = \tilde{N}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$F_0 = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \\ 0 \end{pmatrix}, F_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, F_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Dans le fichier exemple suivant le problème est approché par du P_2 continu pour la vitesse et du P_1 continu pour la pression. Dans ce fichier la prise en compte de l'espace $L_0^2(\Omega)$ (cf. la justification au Paragraphe 2.10.3) se fait par les instructions

`D.inc_point{1,1} = 3 ;`

`D.inc_point{1,2} = 'end' ;`

`D.inc_point{1,3} = 0 ;`

qui permettent de remplacer le troisième espace d'EF de l'espace EF produit E pour l'approximation de la pression qui est le P_1 continu par l'espace P_1 continu privé de sa dernière fonction de base. Une fois calculée, la solution est exprimée dans l'espace tout entier avec le coefficient 0 pour la dernière composante dans la décomposition de la pression sur la base canonique du P_1 continu. On donne ci-après le *script* du fichier de fonction exemple pour ce problème de Stokes.

Fichier de fonction de l'exemple 2 :*EF/exemples/exemple2_EF_assemblage.m*

```

% Fichier EF/exemples/exemple2_EF_assemblage.m
% Probleme de Stokes. Ecoulement de Poiseuille
% Utilisation de EF/EF_assemblage.
% Calcul relatif aux espaces EF.
% Commande
%      cd C :/EF, path_EF, cd C :/EF/exemples
%      [Uc, donnees] = exemple2_EF_assemblage ;

function [Uc, donnees] = exemple2_EF_assemblage ;
close all ;

%Maillage de 8 triangles, chaque triangle a au plus
%  une arete sur la frontiere
T = [1,5,4 ; 1,2,5 ; 3,6,5 ; 2,3,5 ; 4,5,7 ; 7,5,8 ; 5,9,8 ; 5,6,9] ;
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = 9 ;
Front{1} = [7,8,9] ;
Front{2,1} = [9,6,3] ; %mise en colonne
Front{3} = [3,2,1] ;
Front{4} = [1,4,7] ;

%Variable decrivant les espaces d'EF
%Types d'espaces d'EF : F1 = P1, F2 = P2
D.E = [1,2] ;
%Derivation (toujours fixe dans ce code)
D.der = [0,1,2] ;
%Numero de la formule de quadrature sur les triangles
D.intT = 4 ;
%Numero de la formule de quadrature sur les aretes
D.intA = 4 ; %facultatif ici

```

4.3. DESCRIPTION DES FONCTIONS FONCTION EF/EF_assemblage (*Exemple 2*)

```
%Choix de l'espace produit : ici  $E = F_1 \times F_2 \times F_1$ 
D.f = [2,2,1] ;
%Derniere inconnue a supprimer (pression de moyenne nulle)
D.inc_point{1,1} = 3 ;
D.inc_point{1,2} = 'end' ;
D.inc_point{1,3} = 0 ;
%Structure de donnees decrivant le probleme EF
donnees.D = D ;

%Donnees de la viscosite
donnees.eta = 1 ;

%Nom du fichier de maillage
donnees.nom_mail = ...
    'C :/EF/exemples/rep_exemples_mat/exemple2_EF_assemblage_mail' ;

%Sauvegarde des donnees de maillage
save(donnees.nom_mail,'X','Y','T','Noeu','Front')

%Nom du fichier relatif aux EF et aux formules de quadrature
donnees.nom_prep = ...
    'C :/EF/exemples/rep_exemples_mat/exemple2_EF_assemblage_mat' ;

%Conditions aux limites
donnees.fcb = @cb_don ;

%Equations (formes bilineaires) :
donnees.feq_Dom = @eq_Dom ; %sur le domaine

%Seconds membres (formes lineaires)
donnees.fscm_Dom = @scm_Dom ; %sur le domaine

%Utilisation de EF_prepa_don
EF_prepa_don(donnees) ;
```

```

%Initialisation de Uc au vecteur nul de l'espace EF produit
Uc = init_transf_cell('init', [], donnees);
[S,f] = EF_assemblage(donnees,Uc);

%Resolution
U = S\f;

%On remet U sous forme de cellules Uc pour le trace
U = full(U);
Uc = init_transf_cell('ta2c',U,donnees);

%Trace de la solution
load(donnees.nom_mail,'X','Y','T');
trac_solEc(donnees, Uc, 0, 3, 50,1);
trac_solEc(donnees, Uc, 0, [1,2], 10, 2, [1,2]);

%Verifications : comparaison avec le Poiseuille
disp('Erreur en P2-P1 : (1.3323e-015) :');
load(donnees.nom_prep,'Xc','Yc');
erreur = max(abs( Uc{1}' - 4*Yc{2}.*(1 - Yc{2})));
disp(erreur);

%Sous-fonctions :
%Membre de gauche
function [Mc, Nc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, ...
                           c2, de, Ac, Bc, numK);
%Membre de gauche : ecoulement de Stokes.

%Termes intermediaires
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
%Viscosite donnee par l'utilisateur
eta = donnees.eta;

```

```

D = 2*eta.*Un ;
%Matrices : forme factorisee
Mc{1} = {Zero, Zero,-Un ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero,-Un ;
          Zero, Zero, Zero } ;
Mc{2} = {D,      Zero, Zero ;
          Zero, D/2, Zero ;
          Zero, D/2, Zero ;
          Zero, Zero, Zero ;
          -Un,    Zero, Zero } ;
Mc{3} = {Zero, Zero, Zero ;
          D/2,   Zero, Zero ;
          D/2,   Zero, Zero ;
          Zero, D,   Zero ;
          Zero, -Un, Zero } ;
Nc{1} = {Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Un  } ;
Nc{2} = {Un,     Zero, Zero ;
          Zero, Un/2, Zero ;
          Zero, Un/2, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero } ;
Nc{3} = {Zero, Zero, Zero ;
          Un/2, Zero, Zero ;
          Un/2, Zero, Zero ;
          Zero, Un,   Zero ;
          Zero, Zero, Zero } ;

%Second membre

```

4. DESCRIPTION DU CODE

FONCTION EF/EF_assemblage (*Exemple 2*)

```
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, a2, ...
                                b2, c2, de, Ac, Bc, numK) ;

%Second membre :  $f(x,y) = 0$ .

%Termes intermediaires
Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;

%Matrices : forme factorisee
Mc{1} = {Un, Zero, Zero ;
         Zero, Un, Zero ;
         Zero, Zero, Un} ;
Mc{2} = {Zero, Zero, Zero ;
         Zero, Zero, Zero ;
         Zero, Zero, Zero} ;
Mc{3} = Mc{2} ;
Nc = Mc ;

%Second membre f
Fc{1} = {Zero ; Zero ; Zero} ;
Fc{2} = {Zero ; Zero ; Zero} ;
Fc{3} = {Zero ; Zero ; Zero} ;

%Conditions au bord
function [fb,sb] = cb_don(donnees)
%Conditions au bord : Poiseuille en  $x = 0$  et  $x = 1$ .

%Frontieres avec Dirichtlet (pas de conditions sur la pression)
fb{1} = [1 :4] ;
fb{2} = [1 :4] ;
fb{3} = [] ;
%Valeurs des fonctions sur les frontieres
%D'abord mise a zero
for i = 1 :4
```



```

sb.v{1,i} = @f1 ;
sb.v{2,i} = @f1 ;
end ;
%Vitesse en entree :
sb.v{1,4} = @f2 ;
%Vitesse en sortie :
sb.v{1,2} = @f2 ;

%Sous-fonctions pour les conditions au bord
function z = f1(x,y)
z = 0.*x ;
%Entree et sortie
function z = f2(x,y) ;
z = 4*y.*(1 - y) ;

```

Exemple 3.

Le cas de la loi puissance est obtenu remplaçant dans le problème précédent la viscosité constante η par une viscosité qui suit une loi de viscosité de type «loi puissance» en posant ici :

$$\eta(x, y) = |d(u(x, y))|^{r-2} \text{ avec } |d(u)|^2 = \sum_{1 \leq i, j \leq 2} d_{ij}(u) d_{ij}(u)$$

où r est un paramètre pris dans $]1, +\infty[$. Les conditions au bord sont celles de l'écoulement de Poiseuille donné pour la vitesse par

$$u_1(x, y) = 1 - (2|y - 1/2|)^{\frac{r}{r-1}}, u_2(x, y) = 0,$$

avec la pression solution

$$p(x, y) = -2^{1+r/2} \left(\frac{r}{r-1} \right)^{r-1} x.$$

Ce cas est traité dans le fichier *EF/exemples/exemple2r-EF_assemblage.m* qui donne un exemple d'utilisation du code sur une méthode de point fixe et de l'utilisation de l'ancien itéré pour calculer le terme de viscosité. Ici, le terme de viscosité est calculé à l'aide de la variable `uc` passé dans la fonction `eq_Dom` de calcul du premier membre, cette variable contenant les dérivées d'ordre 0 et 1 de la solution aux points de la formule de quadrature, avec :

$$u_{1,1}(m_K) = \text{uc}\{1, 2\}; \quad u_{1,2}(m_K) = \text{uc}\{1, 3\};$$

$$u_{2,1}(m_K) = \text{uc}\{2, 2\}; \quad u_{2,2}(m_K) = \text{uc}\{2, 3\}.$$

Le maillage comptant peu de triangles, la solution n'est pas très précise pour l'approximation P_2^c - P_1^c vitesse-pression, ceci s'améliore nettement avec l'utilisation de l'EF P_3^c pour la vitesse et P_2^c pour la pression, notamment pour la valeur de $r = 3/2$, car alors la solution est dans l'espace EF d'approximation. Ceci est du d'une part au fait que u_1 est alors polynômiale de degré 3 sur $\{y \leq 1/2\} \cap \Omega$ et sur $\{y \geq 1/2\} \cap \Omega$ (mais pas sur Ω) et d'autre part au choix du maillage, avec des triangles soit dans $\{y \leq 1/2\} \cap \Omega$ soit dans $\{y \geq 1/2\} \cap \Omega$. On donne ci-après le *script* du fichier de fonction pour la résolution approchée de ce problème.

Fichier de fonction de l'exemple 3 :

EF/exemples/exemple2r_EF_assemblage.m

```
% Fichier EF/exemples/exemple2r_EF_assemblage.m
% Probleme de fluide-Loi puissance. Ecoulement de Poiseuille.
% LOI PUISSANCE. R = 3/2. Aproximation P3-P2.
% Utilisation de EF/EF_assemblage. Calcul relatif aux espaces EF.
% Calcul relatif aux espaces EF.
% Commande
%      cd C :/EF, path_EF, cd C :/EF/exemples
%      [S, Uc, donnees, err] = exemple2r_EF_assemblage ;

function [S, Uc, donnees, err] = exemple2r_EF_assemblage
disp('Loi puissance, EF P2-P3, r=3/2 (14 iterations de pt fixe)');

%Maillage de 8 triangles, chaque triangle a au plus
%  une arete sur la frontiere.
close all ;
T = [1,5,4 ; 1,2,5 ; 3,6,5 ; 2,3,5 ; 4,5,7 ; 7,5,8 ; 5,9,8 ; 5,6,9] ;
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = 9 ;
Front{1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;%mise en colonne
Front{3} = [3,2,1] ;
Front{4} = [1,4,7] ;
```

```

%Variable decrivant les espaces d'EF
%Types d'espaces d'EF : F1 = P1, F2 = P2
D.E = [2,3] ;
%Derivation (toujours fixe dans ce code)
D.der = [0,1,2] ;
%Numero de la formule de quadrature sur les triangles
D.intT = 10 ;
%Numero de la formule de quadrature sur les aretes
D.intA = 4 ;
%Choix de l'espace produit : ici E = F2 x F2 x E1
D.f = [2,2,1] ;
%Derniere inconnue a supprimer (pression de moyenne nulle)
D.inc_point{1,1} = 3 ;
D.inc_point{1,2} = 'end' ;
D.inc_point{1,3} = 0 ;
%Structure de donnees decrivant le probleme EF
donnees.D = D ;
%Donnees utilisateur : viscosite
donnees.eta = 1 ;

%Nom du fichier de maillage
donnees.nom_mail = ...
    'C :/EF/exemples/rep_exemples_mat/exemple2r_EF_assemblage_mail' ;

%Sauvegarde des donnees de maillages
save(donnees.nom_mail,'X','Y','T','Noeu','Front')

%Nom du fichier de matrices
donnees.nom_prep = ...
    'C :/EF/exemples/rep_exemples_mat/exemple2r_EF_assemblage_mat' ;

%Conditions aux limites
donnees.fcb = @cb_don ;

```

```

%Equations (formes bilineaires) :
donnees.feq_Dom = @eq_Dom; %sur le domaine
%Seconds membres (formes lineaires)
donnees.fscm_Dom = @scm_Dom; %sur le domaine

%Utilisation de EF_prepa_don
EF_prepa_don(donnees);

%Initialisation de Uc au vecteur nul de l'espace EF produit
Uc = init_transf_cell('init', [], donnees);

%Resolution du probleme de Stokes
donnees.r = 2;
[S,f] = EF_assemblage(donnees,Uc);
U = S\f;
U = full(U);
Uc = init_transf_cell('ta2c',U,donnees);

%Iterations de point fixe pour la loi puissance
donnees.r = 3/2;
format loose
for i = 1 :14%erreur oscille apres convergence
    U1 = U;
    [S,f] = EF_assemblage(donnees,Uc);
    %Resolution
    U = S\f;
    %On remet U sous forme de cellules Uc pour le trace
    U = full(U);
    Uc = init_transf_cell('ta2c',U,donnees);
    disp(['Loi puissance, iteration numero ', num2str(i)]);
    disp('erreur entre deux iteres')
    err(i) = norm(U - U1);
    disp(norm(U - U1));
end;

```

```

format

%Trace de la solution
load(donnees.nom_mail,'X','Y','T');
trac_solEc(donnees, Uc, 0, 3, 50,1);
trac_solEc(donnees, Uc, 0, [1,2], 10, 2, [1,2]);

%Verification
disp('Erreur sur la vitesse en P3-P2 : (2.2114e-011) :');
load(donnees.nom_prep,'Xc','Yc');
r = donnees.r;
z = 1 - (2*abs(Yc{2} - 1/2)).^(r/(r - 1));
erreur = max(abs( Uc{1}' - z));
disp(erreur);

disp('Erreur sur la pression en P3-P2 : (1.2570e-006) :');
z = -2^(1 + r/2)*(r/(r - 1))^(r - 1)*Xc{1};
%Mise de la derniere composante a z(end) = 0 en ajoutant
%  une constante :
z = z - z(end);
disp(max(abs(Uc{3} - z')));

%Sous-fonctions :
%Membre de gauche
function [Mc, Nc] = eq_Dom(donnees, uc, ...
    a1, b1, c1, a2, b2, c2, de, Ac, Bc, numK);
%Membre de gauche : equation de Stokes-Loi puissance.

%Termes intermediaires
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));

%Viscosite donnee par l'utilisateur
r = donnees.r;

```

```

u11 = uc{1,2} ;
u12 = uc{1,3} ;
u21 = uc{2,2} ;
u22 = uc{2,3} ;
d11 = u11 ;
d12 = (u12 + u21)/2 ;
d21 = d12 ;
d22 = u22 ;
du = (d11.*d11 + d12.*d12 + d21.*d21 + d22.*d22).^(1/2) ;
D = 2*(donnees.eta)*du.^(r - 2) ;

%Matrices : forme factorisee
Mc{1} = {Zero, Zero, -Un ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, -Un ;
          Zero, Zero, Zero } ;
Mc{2} = {D,      Zero, Zero ;
          Zero, D/2, Zero ;
          Zero, D/2, Zero ;
          Zero, Zero, Zero ;
          -Un,   Zero, Zero } ;
Mc{3} = {Zero, Zero, Zero ;
          D/2,   Zero, Zero ;
          D/2,   Zero, Zero ;
          Zero, D,   Zero ;
          Zero, -Un, Zero } ;
Nc{1} = {Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Zero ;
          Zero, Zero, Un  } ;
Nc{2} = {Un,    Zero, Zero ;
          Zero, Un/2, Zero ;

```

```

        Zero, Un/2, Zero ;
        Zero, Zero, Zero ;
        Zero, Zero, Zero } ;
Nc{3} = {Zero, Zero, Zero ;
        Un/2, Zero, Zero ;
        Un/2, Zero, Zero ;
        Zero, Un,   Zero ;
        Zero, Zero, Zero } ;

%Second membre
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, a2, ...
                                b2, c2, de, Ac, Bc, numK) ;

%Second membre : f(x,y) = 0.

%Termes intermediaires
Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;

%Matrices : forme factorisee
Mc{1} = {Un, Zero, Zero ;
        Zero, Un, Zero ;
        Zero, Zero, Un} ;
Mc{2} = {Zero, Zero, Zero ;
        Zero, Zero, Zero ;
        Zero, Zero, Zero} ;
Mc{3} = Mc{2} ;
Nc = Mc ;

%Second membre f
Fc{1} = {Zero ; Zero ; Zero} ;
Fc{2} = {Zero ; Zero ; Zero} ;
Fc{3} = {Zero ; Zero ; Zero} ;

%Conditions au bord

```

```

function [fb,sb] = cb_don(donnees)
%Conditions au bord : Poiseuille en x = 0 et x = 1.

%Frontieres avec Dirichtlet (pas de conditions sur la pression)
fb{1} = [1 :4] ;
fb{2} = [1 :4] ;
fb{3} = [] ;
global r
r = donnees.r ;
%Valeurs des fonctions sur les frontieres
%D'abord mise a zero
for i = 1 :4
sb.v{1,i} = @f1 ;
sb.v{2,i} = @f1 ;
end ;
%Vitesse en entree
sb.v{1,4} = @f2 ;
%Vitesse en sortie
sb.v{1,2} = @f2 ;

%Sous-fonctions pour les conditions au bord
function z = f1(x,y)
z = 0.*x ;
%Entree et sortie
function z = f2(x,y) ;
global r ;
z = 1 - (2*abs(y - 1/2)).^(r/(r - 1)) ;

```

Exemple 4.

Le problème de la cavité entraînée est traité par le fichier de fonction :

EF/exemples/exemple3_EF_assemblage.m.

Dans ce fichier on a utilisé, à titre d'exemple, une représentation non factorisée de la forme bilinéaire dans le cas d'un système. Cette représentation est de la forme :

$$\int_{\Omega} \left(\sum_{0 \leq i, j \leq 2} (M_{ij} U_{,j}, N_{ij} V_{,j}), \right.$$

avec

$$U = \begin{pmatrix} u \\ p \end{pmatrix}_{,i}, V = \begin{pmatrix} v \\ q \end{pmatrix}_{,i}.$$

A partir de la relation

$$\begin{aligned} 2\eta(d(u), d(v)) - (p, \nabla \cdot v) = & 2\eta\left(u_{1,1}v_{1,1} + \frac{1}{2}(u_{1,2}v_{1,2} + u_{1,2}v_{2,1} + u_{2,1}v_{1,2} + u_{2,1}v_{2,1}) \right. \\ & \left. + u_{2,2}v_{2,2}\right) - pv_{1,1} - pv_{2,2} \\ -(\nabla \cdot u, q) = & -u_{1,1}q - u_{2,2}q \end{aligned}$$

on obtient, avec $D = 2\eta$, les valeurs des matrices :

$$\begin{aligned} M_{00} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{11} = \begin{pmatrix} D & 0 & 0 \\ 0 & D/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{22} = \begin{pmatrix} D/2 & 0 & 0 \\ 0 & D & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ M_{21} &= \begin{pmatrix} D/2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & D/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ M_{01} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}; M_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}; \\ M_{10} &= \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{20} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ N_{00} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{11} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{22} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ N_{21} &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ N_{01} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}; N_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}; \\ N_{10} &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{20} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

Sinon, le fichier d'exemple s'obtient à partir du fichier de l'exemple 2 pour l'écoulement de Poiseuille en modifiant simplement (outre la fonction `eq_Dom` pour l'illustration ci-dessus) la fonction de condition au bord en prenant pour u_0 une vitesse nulle au bord sauf sa première composante sur $x = 0$: $u_{01}(0, y) = 1$ qui donne la vitesse d'entraînement. On donne ci-après le fichier pour ce problème.

Fichier de fonction de l'exemple 4 :*EF/exemples/exemple3_EF_assemblage.m*

```

% Fichier EF/exemples/exemple3_EF_assemblage.m
% CAVITE ENTRAINÉE, Stokes.
% Utilisation de EF/EF_assemblage.
% Calcul relatif aux espaces EF.
% Commande
%      cd C :/EF, path_EF, cd C :/EF/exemples
%      [Uc, donnees] = exemple3_EF_assemblage;

function [S, Uc, donnees] = exemple3_EF_assemblage
close all;
disp('Cavite entrainee');
%Maillage de 8 triangles, chaque triangle a au plus
%  une arete sur la frontiere
T = [1,5,4; 1,2,5; 3,6,5; 2,3,5; 4,5,7; 7,5,8; 5,9,8; 5,6,9];
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1];
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1];
Noeu = 9;
Front{1} = [7,8,9];
Front{2,1} = [9,6,3]; %mise en colonne
Front{3} = [3,2,1];
Front{4} = [1,4,7];

%Variable decrivant les espaces d'EF
%Types d'espaces d'EF : F1 = P1, F2 = P2
D.E = [1,2];
%Derivation (toujours fixe dans ce code)
D.der = [0,1,2];

%Numero de la formule de quadrature sur les triangles
D.intT = 4;

```

4.3. DESCRIPTION DES FONCTIONS FONCTION EF/EF_assemblage (*Exemple 4*)

```
%Numero de la formule de quadrature sur les aretes
D.intA = 4 ;

%Choix de l'espace produit : ici  $E = F_1 \times F_2 \times E_1$ 
D.f = [2,2,1] ;

%Derniere inconnue a supprimer (pression de moyenne nulle)
D.inc_point{1,1} = 3 ;
D.inc_point{1,2} = 'end' ;
D.inc_point{1,3} = 0 ;

%Structure de donnees decrivant le probleme EF
donnees.D = D ;

%Nom du fichier de maillage
donnees.nom_mail = ...
    'C :/EF/exemples/rep_exemples_mat/exemple3_EF_assemblage_mail' ;

%Sauvegarde des donnees de maillages
save(donnees.nom_mail,'X','Y','T','Noeu','Front')

%Nom du fichier de matrices
donnees.nom_prep = ...
    'C :/EF/exemples/rep_exemples_mat/exemple3_EF_assemblage_mat' ;

%Conditions aux limites
donnees.fcb = @cb_don ;

%Equations (formes bilineaires) :
donnees.feq_Dom = @eq_Dom ; %sur le domaine

%Seconds membres (formes lineaires)
donnees.fscm_Dom = @scm_Dom ; %sur le domaine
```

```

%viscosite
donnees.eta = 1;

%Utilisation de EF_prepa_don
EF_prepa_don(donnees);

%Initialisation de Uc au vecteur nul de l'espace EF produit
Uc = init_transf_cell('init', [], donnees);

[S,f] = EF_assemblage(donnees,Uc);

%Resolution
U = S\f;

%On remet U sous forme de cellules Uc pour le trace
U = full(U);
Uc = init_transf_cell('ta2c',U,donnees);

%Trace de la solution
load(donnees.nom_mail,'X','Y','T');
trac_solEc(donnees, Uc, 0, [], 4, 1, [1,2]);

%Sous-fonctions :
%Membre de gauche
function [Mc, Nc] = eq_Dom(donnees, uc, ...
    a1, b1, c1, a2, b2, c2, de, Ac, Bc, numK);
%Membre de gauche : ecoulement de Stokes.

%Termes intermediaires
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));

%Viscosite donnee par l'utilisateur
eta = donnees.eta;

```

```

D = 2*eta.*Un ;

%Matrices : forme non factorisee
for i = 1 :3
    for j = 1 :3
        for k = 1 :3
            for m = 1 :3
                Mc{i,j}{k,m} = Zero ;
                Nc{i,j}{k,m} = Zero ;
            end ;
        end ;
    end ;
end ;

%u1,1*v1,1
Mc{2,2}{1,1} = D ;
Nc{2,2}{1,1} = Un ;
%1/2*u2,1*v2,1
Mc{2,2}{2,2} = D/2 ;
Nc{2,2}{2,2} = Un ;
%1/2*u1,2*v2,1
Mc{3,2}{1,1} = 1/2*D ;
Nc{3,2}{1,2} = Un ;
%1/2*u2,1*v1,2
Mc{2,3}{2,2} = 1/2*D ;
Nc{2,3}{2,1} = Un ;
%1/2*u1,2*v1,2
Mc{3,3}{1,1} = D/2 ;
Nc{3,3}{1,1} = Un ;
%u2,2*v2,2
Mc{3,3}{2,2} = D ;
Nc{3,3}{2,2} = Un ;
%pressions
Mc{1,2}{3,3} = -Un ;
Nc{1,2}{3,1} = Un ;

```

```

Mc{1,3}{3,3} = -Un ;
Nc{1,3}{3,2} = Un ;
Mc{2,1}{1,1} = -Un ;
Nc{2,1}{1,3} = Un ;
Mc{3,1}{2,2} = -Un ;
Nc{3,1}{2,3} = Un ;

%Second membre
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, a2, ...
                                b2, c2, de, Ac, Bc, numK) ;

%Second membre : f(x,y) = 0.

%Termes intermediaires
Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;

%Matrices
Mc{1} = {Un, Zero, Zero ;
         Zero, Un, Zero ;
         Zero, Zero, Un} ;
Mc{2} = {Zero, Zero, Zero ;
         Zero, Zero, Zero ;
         Zero, Zero, Zero} ;
Mc{3} = Mc{2} ;
Nc = Mc ;

%Second membre f
Fc{1} = {Zero ; Zero ; Zero} ;
Fc{2} = {Zero ; Zero ; Zero} ;
Fc{3} = {Zero ; Zero ; Zero} ;

%Conditions au bord
function [fb,sb] = cb_don(donnees)
%Conditions au bord.

```

4.3. DESCRIPTION DES FONCTIONS FONCTION EF/EF_assemblage (*Exemple 4*)

```
%Entrainement tangentiel en  $x = 0$ , vitesse nulle sur les
%   autres frontieres.

%Frontieres avec Dirichtlet (pas de conditions sur la pression)
fb{1} = [1 :4] ;
fb{2} = [1 :4] ;
fb{3} = [] ;
%Valeurs des fonctions sur les frontieres
%D'abord mise a zero
for i = 1 :4
    sb.v{1,i} = @f1 ;
    sb.v{2,i} = @f1 ;
end ;
%Vitesse en entree
sb.v{2,4} = @f2 ;

%Sous-fonctions pour les conditions au bord
function z = f1(x,y)
    z = 0.*x ;
%Entree et sortie
function z = f2(x,y) ;
    z = y.^0 ;
```

Fonctions de EF/deg_lib.**Fonction :**

```
function a = triangle_arete(T) ;
```

Tableau de connectivité des arêtes du maillage de numéros de sommets donnés par le tableau de trois colonnes T.

Variables d'entrée :

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

a : SPARSE ARRAY. Matrice telle que $a(i, j) = 1$ si le segment de sommets i, j , avec $i < j$, appartient à un triangle du maillage. Les autres termes de a sont nuls. En particulier a est triangulaire supérieure.

Exemple : Cet exemple est donné dans le `help` de la fonction `triangle_arete`, et porte sur un maillage de deux triangles. La figure 4.3-2 ci-après représente un maillage ayant les numéros de sommets de cet exemple.

EXEMPLE

```
T=[1,2,3 ;1,3,4] ;
a=triangle_arete(T) ;
disp(T) ;
disp(full(a))
```

On obtient pour la matrice a la valeur suivante :

$$a = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Par exemple, en regardant uniquement la partie triangulaire supérieure stricte de a, le terme $a(1, 2) = 1$ montre que le segment de sommets 1 et 2 est une arête du maillage, le terme $a(2, 4) = 0$ indique que le segment de sommets numéros 2 et 4 n'est pas un arête du maillage.

Algorithme :

EF/deg_lib/triangle_arete.m

On remet ici la partie déclarative de la fonction :

```
function a = triangle_arete(T) ;
```

- Pour obtenir une matrice **a** triangulaire supérieure, on ordonne les lignes de **T** :

```
t = sort(T,2) ;
```

- Initialisation de **a** matrice de type **SPARSE** à la matrice carrée nulle de taille donnée par le nombre de sommets du maillage.

- Boucle sur les triangles. On pose $a(k, m) = 1$ pour (m, k) numéro de sommets du triangle courant numéro i tel que $m < k$.

```
Pour i=1 :size(t,1)
```

```
    %On a necessairement i1<i2<i3 avec le choix de t
```

```
    i1 = t(i,1) ;
```

```
    i2 = t(i,2) ;
```

```
    i3 = t(i,3) ;
```

```
    a(i1,i2) = 1 ;
```

```
    a(i1,i3) = 1 ;
```

```
    a(i2,i3) = 1 ;
```

```
Fin ;
```

Fonction :

```
function [A, B, XA, YA] = maillage_triangle_arete(T, X, Y) ;
```

Numérotation globale des arêtes du maillage de sommets décrits par T et correspondance entre les numéros d'arêtes et les numéros de triangles qui les porte, coordonnées des points milieux des arêtes en suivant l'ordre choisi pour les arêtes. Les variables XA et YA sont les coordonnées cartésiennes des points situés sur les milieux d'arêtes. L'arête de numéro i , dans la numérotation globale des arêtes, a pour point milieu le point de coordonnées $(XA(i), YA(i))$. La numérotation des arêtes est utilisée en particulier pour la construction des *ddl* situés sur les arêtes. Ainsi, l'ordre de numérotation choisi pour les *ddl* milieux d'arête de l'espace d'EF P_2^c sera aussi celui des arêtes, les coordonnées cartésiennes de ces *ddl* sont alors donnés, dans l'ordre, par XA et YA. Pour le P_3^c , on numérottera les deux *ddl* de l'arête numéro 1, avec l'ordre indiqué dans la description de la fonction `maillage_ddl`, puis les deux *ddl* de l'arête numéro 2, *etc.*, on termine ensuite en numérotant les *ddl intérieurs* dans l'ordre des triangles.

Pour la suite, rappelons la numérotation locale des arêtes adoptée au Paragraphe 2.3. Les sommets de chaque triangle numéro k du maillage ont une numérotation de 1 à 3 dite locale définie par leur ordre dans $T(k, :)$. Pour un triangle, de sommets a_1, a_2, a_3 dans l'ordre de numérotation locale, on numérottera localement les arêtes de ce triangle de la manière suivante : la première arête est le segment d'extrémités a_1, a_2 , la seconde celle d'extrémités a_1, a_3 et la troisième a_2, a_3 .

Variables d'entrée :

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

X, Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

A : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes du maillage) $\times$ 4. Matrice de correspondance entre les numéros d'arêtes et les numéros de triangles. $A(i, 1)$ est le numéro de triangle auquel appartient l'arête n° i , $A(i, 2)$ le numéro local de cette arête sur ce triangle. Même chose pour $A(i, 3)$ et $A(i, 4)$ si l'arête est portée par un second triangle, dans le cas contraire ces deux derniers termes prennent la valeur 0.

B : DOUBLE ARRAY. Matrice de taille (Nombre de triangles sur la frontière) $\times$ 3. Matrice de correspondance entre les triangles et les arêtes. $B(i, k)$ est le numéro d'arête dans la numérotation globale des arêtes de l'arête n° k du triangle i , dans la numérotation locale des arêtes de ce triangle ($k = 1, 2, 3$).

XA, YA : DOUBLE ARRAY. Vecteurs lignes des coordonnées cartésiennes des milieux d'arête. $XA(i), YA(i)$, sont les coordonnées du point au milieu de l'arête n° i .

4.3. DESCRIPTION DES FONCTIONS FONCTION `deg_lib/maillage_triangle_arete`

Exemple : Cet exemple est donné dans le `help` de la fonction `maillage_triangle_arete` et reprend le maillage par deux triangles de l'exemple précédent, en précisant ici le choix coordonnées. Cet exemple renvoie le graphique du maillage avec les indications des sommets ($\circ$ bleu) et des points milieux ($\diamond$ rouges) (*cf.* la Figure 4.3-2 ci-après).

EXEMPLE

```
T=[1,2,3;1,3,4] ;
X=[0,0,1,1] ;
Y=[1,0,0,1] ;
[A,B,XA,YA] = maillage_triangle_arete(T,X,Y) ;
hold on ;
patch(X(T(1, :)),Y(T(1, :)),'w') ;
patch(X(T(2, :)),Y(T(2, :)),'w') ;
plot(X,Y,'o') ;
plot(XA,YA,'rd') ;
```

La représentation graphique du maillage de l'exemple, de ses sommets et de la numérotation des milieux d'arête obtenue est donnée dans la figure 4.3-2 suivante.

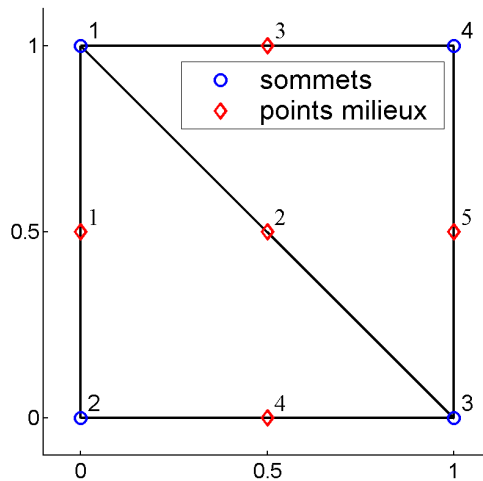


Figure 4.3-2. Numéros des sommets et des points milieu des arêtes du maillage.

Les matrices A et B prennent les valeurs suivantes :

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 2 & 1 & 1 & 2 \\ 2 & 2 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 2 & 3 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \end{pmatrix}.$$

Par exemple le terme $A(1, [1, 2]) = (1, 1)$ indique que l'arête numéro 1 (en numérotation globale) appartient au triangle numéro 1 et que son numéro d'arête en numérotation locale sur ce triangle vaut 1, c'est donc l'arête de sommets $T(1, [1, 2]) = (1, 2)$. Les termes nuls $A(1, [3, 4]) = (0, 0)$ indiquent que cette arête est placée sur la frontière du domaine. Le terme $A(2, [1, 2]) = (2, 1)$ indique que l'arête numéro 2 (en numérotation globale) est sur le triangle 2 et que son numéro local d'arête sur ce triangle vaut 1, elle a donc pour sommets $T(2, [1, 2]) = (1, 3)$, ensuite le terme $A(2, [3, 4]) = (1, 2)$ indique que cette arête est aussi sur le triangle 1 avec la valeur 2 de son numéro local d'arête sur ce triangle.

La première ligne de B : $B(1, :) = (1, 2, 4)$ indique que les trois arêtes successives (en numérotation locale) du triangle 1 sont les arêtes (en numérotation globale) 1, 2 et 4, la seconde ligne $B(2, :) = (2, 3, 5)$ indique que le triangle numéro 2 est constitué des arêtes (en numérotation globale) 2, 3 et 5.

Algorithme :

EF/deg_lib/maillage_triangle_arete.m

On remet ici la partie déclarative de la fonction :

```
function [A, B, XA, YA] = maillage_triangle_arete(T, X, Y);
```

- Calcul du tableau de connectivité des arêtes.

```
    a = triangle_arete(T);
```

- Les arêtes sont numérotées en parcourant les lignes de a ce qui revient à parcourir les colonnes de $c = a'$.

```
    c = a';
```

```
    I = find(c~=0);
```

```
    %Numerotation des aretes de 1 a length(I) :
```

```
    c(I) = 1 : length(I);
```

```
    a = c';
```

```
    a = (a' + a);
```

- Ci-dessus on a mis dans a la numérotation des arêtes. Le terme $a(i, j)$ est alors le numéro global d'arête de l'arête formée des sommets n^{os} i et j . Remarque : $\text{diag}(a) = 0$ (vecteur nul).

- Boucle sur les triangles, pour le calcul de B et de XA, YA à partir de a et des données.

Ci-dessous $i1, i2, i3$ sont les numéros de sommets du triangle i . Ensuite, $a(i1, i2)$, $a(i1, i3)$ et $a(i2, i3)$ sont les numéros d'arêtes, dans la numérotation globale, des arêtes formées des sommets numéros $(i1, i2)$, $(i1, i3)$ et $(i2, i3)$. Localement, la première arête sur le triangle i est choisi comme l'arête de sommets numéros $(i1, i2)$, la seconde arête de sommets numéros $(i1, i3)$ et la troisième de sommets de numéros $(i2, i3)$.

4.3. DESCRIPTION DES FONCTIONS FONCTION `deg_lib/maillage_triangle_arete`

- Initialisation de B, XA, YA à leur taille

```
B = zeros([size(T,1),3]) ;
```

```
XA = zeros(1,length(I)) ;
```

```
YA = zeros(1,length(I)) ;
```

- Boucle

```
Pour i=1 :size(T,1)
```

```
    %Choix de la numerotation locale des aretes
```

```
    i1 = T(i,1) ;
```

```
    i2 = T(i,2) ;
```

```
    i3 = T(i,3) ;
```

```
    B(i,1 :3) = [a(i1,i2),a(i1,i3),a(i2,i3)] ;
```

```
    %Calcul des coordonnees des points milieux. Ci-dessous
```

```
    %   XA et YA ont pu deja etre calculees car on passe
```

```
    %   deux fois sur une arete commune a deux triangles.
```

```
    %   Le resultat est inchange.
```

```
    XA(a(i1,i2)) = (X(i1) + X(i2))/2 ;
```

```
    YA(a(i1,i2)) = (Y(i1) + Y(i2))/2 ;
```

```
    XA(a(i1,i3)) = (X(i1) + X(i3))/2 ;
```

```
    YA(a(i1,i3)) = (Y(i1) + Y(i3))/2 ;
```

```
    XA(a(i2,i3)) = (X(i2) + X(i3))/2 ;
```

```
    YA(a(i2,i3)) = (Y(i2) + Y(i3))/2 ;
```

Fin

- Boucle sur les triangles, pour le calcul de A à partir de B.

- Initialisation de A à sa taille

```
A = zeros(max(max(B)),4) ;
```

Ci-dessous, pour $j = 1, 2, 3$, $B(i, j)$ est la valeur du numéro global de l'arête du triangle numéro i qui a pour numéro j dans la numérotation locale des arêtes sur ce triangle.

- Ensuite, la variable I, avec $[I, J] = \text{find}(B == B(i, j))$ désigne les numéros de lignes de B, et donc les numéros de triangles, qui contiennent le numéro global d'arête $B(i, j)$. Ainsi, l'arête numéro $B(i, j)$ appartient aux triangles de numéros donnés par $I(k)$, $1 \leq k \leq \text{length}(I)$. La ligne $B(i, j)$ de A décrit l'arête numéro $B(i, j)$ et $A(B(i, j), [1, 3])$ prend alors les valeurs de ces numéros de triangles avec $A(B(i, j), 3) = 0$ lorsque l'arête $B(i, j)$ appartient à un unique triangle (pour $\text{length}(I) = 1$).

- Enfin, $A(B(i, j), [2, 4])$ prend avec J, $J(k)$, $1 \leq k \leq \text{length}(I)$ désignant le numéro de colonne de B et donc le numéro local de l'arête $B(i, j)$ sur le triangle $I(k)$, les valeurs les numéros local de l'arête sur chacun des triangle. On pose $A(B(i, j), 4) = 0$ lorsque l'arête $B(i, j)$ appartient à un unique triangle (pour $\text{length}(I) = 1$).

- Boucle

```
Pour i=1 :size(T,1)
```

```
    Pour j=1 :3
```

```
        %Numeros de triangles portant l'arete B(i,j) :
```

```
        [I,J] = find(B==B(i,j)) ;
```

```
        %Cas ou l'arete est commune a deux triangles :
```

```
        Si length(I)==2
```

```
            A(B(i,j),[1,3]) = [I'] ;
```

```
            A(B(i,j),[2,4]) = [J'] ;
```

```
            %Cas ou l'arete appartient a un seul triangle :
```

```
        Sinon
```

```
            A(B(i,j),1 :2) = [I,J] ;
```

```
        Fin ;
```

```
    Fin ;
```

```
Fin ;
```

Fonction :

```
function [Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, ...
                                                    Y, Front) ;
```

Numérotation globale des arêtes du maillage et liste des arêtes du maillage de sommets décrits par **T** (variable **Am**), correspondance entre les arêtes et les triangles (variable **Pa**). Numérotation des arêtes propre à chaque morceau de frontière décrit par **Front**, correspondance de ces arêtes avec leur numéro global d'arête et leurs triangles d'appartenance (variables **Pf** et **Pfc**). Les variables **Pa**, **Pfc** et **Pf** sont en particulier destinées à être stockées dans le fichier relatif aux types d'espaces d'EF choisis et au choix des formules d'intégration du Tableau 2.2.3-1b du Paragraphe 2.2.3.

Cette fonction numérote les arêtes du maillage en utilisant la fonction **maillage_triangle_arete**. Les variables **T**, **X**, **Y** décrivent le maillage du domaine et **Front** une partition de la frontière du domaine en morceaux de frontière. Cette fonction renvoie aussi le tableau **Am**, qui donne arête par arête les numéros de sommets de chaque arête du maillage. Cette fonction établit, avec **Pa**, une correspondance entre le numéro global de chaque arête et le numéro du (ou des) triangle(s) sur lequel elle est placée, en précisant aussi sa numérotation locale sur chaque triangle d'appartenance. Sur chaque morceau de frontière i , les arêtes reçoivent une seconde numérotation. L'ordre choisi est alors celui des numéros de sommets du vecteur **Front** $\{i\}$ (les termes consécutifs de **Front** $\{i\}$ sont nécessairement des numéros de sommets consécutifs sur la frontière du maillage). Enfin, les variables **Pf** et **Pfc** établissent la correspondance pour chaque arête, entre son numéro dans la numérotation locale sur chaque morceau de frontière, son numéro global d'arête et son numéro du (ou des) triangle sur lequel elle est placée, avec sa position locale sur ce triangle.

Variables d'entrée :

Ce sont les variables de maillage décrites au Tableau 2.2.2-1, Paragraphe 2.2.2.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

X,Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Front : CELL ARRAY. Vecteur colonne. La cellule i de **Front** est un vecteur ligne de type DOUBLE ARRAY qui décrit consécutivement sur la frontière les numéros de sommets du morceau de frontière numéro i , *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

Am : **DOUBLE ARRAY**. Matrice de taille (Nombre d'arêtes du maillage) $\times$ 2. Description arête par arête des sommets du maillage : $\text{Am}(i, :)$ désigne les numéros des deux sommets de l'arête numéro i du maillage. L'arête n'est pas orientée. On a la relation $\text{Am}(i, :) = \text{T}(\text{Pa}(i, 1), z)$ avec $z = [1, 2]$ si $\text{Pa}(j, 2) = 1$, $z = [1, 3]$ si $\text{Pa}(j, 2) = 2$, et $z = [2, 3]$ si $\text{Pa}(j, 2) = 3$.

Pa : **DOUBLE ARRAY**. Matrice de taille (Nombre d'arête du maillage) $\times$ 5. Description des arêtes du maillage tel que $\text{Pa}(j, 1)$ est un numéro de triangle auquel appartient l'arête j et $\text{Pa}(j, 2)$ son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour $\text{Pa}(j, 3 : 4)$ si l'arête appartient à un deuxième triangle sinon $\text{Pa}(j, 3)$ et $\text{Pa}(j, 4)$ sont nuls (arête sur la frontière). Enfin $\text{Pa}(j, 5)$ est le numéro du morceau de frontière si l'arête j est sur la frontière, nul sinon ou si on a posé **Front** = []. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Pfc : **CELL ARRAY**. Vecteur colonne de cellules de longueur égale au nombre de morceaux de frontière du domaine. Cette variable a donc le même nombre de cellules que la variable de maillage **Front** (*cf.* Tableau 2.2.2-1, Paragraphe 2.2.2). Sur chaque morceau de frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur ce morceau de frontière. La cellule $\text{Pfc}\{i\}$ est un tableau de trois colonnes de type **DOUBLE ARRAY** qui décrit les arêtes du morceau de frontière numéro i de la manière suivante. Soit une arête de numéro j dans sa numérotation sur le morceau de frontière numéro i , alors $\text{Pfc}\{i\}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $\text{Pfc}\{i\}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3) et $\text{Pfc}\{i\}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes. **Pf** vaut [] si **Front** = []. Le calcul de cette variable est réalisé par `maillage_triangle_arete_ps` à partir de **Front** dans l'ordre des arêtes de **Front**{1}, **Front**{2} *etc.* Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Pf : **DOUBLE ARRAY**. Matrice de taille (Nombre d'arêtes sur la frontière) $\times$ 4. Sur la frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur la frontière. Soit une arête de numéro j dans sa numérotation sur le morceau la frontière, $\text{Pf}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $\text{Pf}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3), $\text{Pf}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes et $\text{Pf}(j, 4)$ le numéro de morceau de frontière auquel appartient cette arête. **Pfc** vaut [] si **Front** = []. Les variables **Pf** et **Pfc** contiennent la même information si l'on sait que $\text{Pf}(:, 1 : 3) = [\text{Pfc}\{1\}; \dots; \text{Pfc}\{\text{end}\}]$. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Remarque. Dans le cas particulier où **Front** représente une partition d'une partie seulement de la frontière (le code n'est pas mis au point pour ce cas), alors pour une arête numéro i , située sur la frontière du domaine qui n'appartient pas à cette partition, la fonction `maillage_triangle_arete_ps` renvoie la valeur $\text{Pa}(i, 5) = 0$, comme pour une arête intérieure. D'autre part cette arête ne figurera ni dans **Pf**

ni dans `Pfc`. On peut poser `Front = []` (cf. `Pa`, `Pfc`, `Pf` ci-dessus), dans ce cas la colonne `Pa(:, 5)` est toujours nulle et on a posé dans ce cas particulier `Pfc = Pf = []`.

Exemple : Cet exemple est donné dans le `help` de la fonction `maillage_triangle_arete_ps` et reprend le maillage par deux triangles de l'exemple précédent. On considère successivement deux partitions de la frontière : partition en un seul morceau et partition en deux morceaux où le premier morceau est le contour du carré privé du côté porté par la droite $y = 0$.

EXEMPLE

```
T=[1,2,3;1,3,4] ;
X=[0,0,1,1] ;
Y=[1,0,0,1] ;
clear('Front')
%Premier exemple de partition de la frontiere
Front{1}=[1,2,3,4,1] ;
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, ...
                                                Y, Front) ;

%Second exemple de partition de la frontiere
Front{1}=[2,1,4,3] ;
Front{2,1}=[2,3] ;
[Am2, Pa2, Pfc2, Pf2] = maillage_triangle_arete_ps(T, X, ...
                                                    Y, Front) ;
```

On obtient pour les matrices `Am = Am2` qui ne dépendent pas du choix de `Front` :

$$Am = Am2 = \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}.$$

Ci-dessus `Am(1, :)` montre que les extrémités de l'arête numéro 1 sont les sommets numéros 1 et 2 du maillage, celles de l'arête numéro 2, numéros 1 et 3 *etc..*

Ci-dessous, la matrice formée des 4 premières colonnes de `Pa` ou bien de `Pa2` est la matrice `A` vue dans l'exemple de la fonction précédente qui relie les arêtes à leurs triangle(s) d'appartenance. La cinquième colonne donne le morceau de frontière auquel appartiennent les arêtes, avec la valeur 0 pour les arêtes intérieures. Pour la première partition de la frontière, `Pa` indique que toutes les arêtes sont sur la frontière 1 à l'exception de l'arête numéro 2 qui est une arête intérieure au domaine. Pour la seconde partition de la frontière, la cinquième colonne de `Pa2` indique que

les arêtes numéro 1, 2 et 3 appartiennent au morceau de frontière numéro 1 et que l'arête numéro 4 est sur le morceau de frontière numéro 2.

$$Pa = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 2 & 0 \\ 2 & 2 & 0 & 0 & 1 \\ 1 & 3 & 0 & 0 & 1 \\ 2 & 3 & 0 & 0 & 1 \end{pmatrix}, \quad Pa2 = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 1 & 2 & 0 \\ 2 & 2 & 0 & 0 & 1 \\ 1 & 3 & 0 & 0 & 2 \\ 2 & 3 & 0 & 0 & 1 \end{pmatrix}.$$

La matrice `Pf` donne la numérotation des arêtes sur la frontière. Par exemple la première ligne de `Pf` indique que l'arête numéro 1 dans la numérotation sur la frontière appartient au triangle 1 avec un numéro local (sur ce triangle) d'arête égal à 1 et un numéro global d'arête de 1, la troisième ligne indique que l'arête numéro 3 dans la numérotation sur la frontière appartient au triangle 2 avec un numéro local (sur ce triangle) d'arête égal à 3 et un numéro global d'arête de 5. La quatrième colonne de `Pf` indique que les arêtes appartiennent toutes au morceau de frontière numéro 1.

$$Pf = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 4 & 1 \\ 2 & 3 & 5 & 1 \\ 2 & 2 & 3 & 1 \end{pmatrix}, \quad Pf2 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 3 & 1 \\ 2 & 3 & 5 & 1 \\ 1 & 3 & 4 & 2 \end{pmatrix}.$$

Enfin `Pfc` et `Pfc2` reprennent les matrices précédentes avec une disposition par cellules :

$$Pfc\{1\} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 3 & 4 \\ 2 & 3 & 5 \\ 2 & 2 & 3 \end{pmatrix}, \quad Pfc2\{1\} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 3 \\ 2 & 3 & 5 \end{pmatrix}, \quad Pfc2\{1\} = (1 \quad 3 \quad 4).$$

Algorithme :

`EF/deg_lib/maillage_triangle_arete_ps.m`

On remet ici la partie déclarative de la fonction :

```
function [Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, ...
                                                    Y, Front);
```

- Numérotation globale des arêtes *cf.* `maillage_triangle_arete` :

```
[A,B,XA,YA] = maillage_triangle_arete(T,X,Y);
```

Les valeurs de `B`, `XA` et `YA` ne sont pas utilisées dans la suite, les résultats sont indépendants de `X` et `Y`. Le terme `A` est la matrice de correspondance entre les numéros d'arêtes et leur triangle d'appartenance.

- Boucle sur les arêtes pour la construction de `Am` à partir de `A` et de `T`. Ci-dessous `i` est le numéro de l'arête courante, `r` son numéro local du triangle numéro `A(i,1)`

4.3. DESCRIPTION DES FONCTIONS FN deg_lib/maillage_triangle_arete.ps

auquelle elle appartient. Ses sommets sont donnés par $T(A(i, 1), [1, 2])$ où $z = [1, 2]$ si $r = 1$, $z = [1, 3]$ si $r = 2$ et $z = [2, 3]$ si $r = 3$.

```
Pour i=1 :size(A,1)
    %r prend les valeurs 1, 2 ou bien 3 :
    r = A(i,2) ;
    %pour r = 1 on a z = [1,2], pour r = 2, z = [1,3] et
    % pour r = 3, z = [2,3] :
    z = (r-3)*(r-2)*[1,2]/2 + (3-r)*(r-1)*[1,3] + ...
        (r-1)*(r-2)*[2,3]/2 ;
    %Une arete commune peut avoir deux orientations,
    % ici ca ne joue pas car on ne rajoute pas de ddl :
    Am(i, :) = [T(A(i,1),z)] ;
Fin ;
%Valeur des quatre premieres colonnes de Pa :
Pa = A ;
```

- Boucle sur les morceaux de frontière décrits par $\text{Front}\{i\}$, dans cette boucle on parcourt les arêtes de $\text{Front}\{i\}$ pour déterminer leur numéro globale d'arête. On en déduit Pf.

Ci-dessous $z = \text{Front}\{i\}([j, j + 1])$ désigne les numéros des deux sommets de la j -ème arête du morceau de frontière i . Pour cette arête on détermine la ligne k de Am qui contiennent ces deux numéros : on a alors $\text{Am}(k, [1, 2]) = z$ ou bien $\text{Am}(k, [2, 1]) = z$. La valeur de k est alors le numéro d'arête globale cherché et $\text{arb}(j, :) = [\text{Pa}(k, 1), \text{Pa}(k, 2), k]$ donne alors la valeur de $\text{Pfc}\{i\}(j, :)$ qui est le numéro du triangle d'appartenance de l'arête frontière k , son numéro local sur le triangle et son numéro dans la numérotation globale de toutes les arêtes.

```
Pour i=1 :length(Front)
    %A la fin de la boucle on aura Pfc{i}=arb
    arb = [] ;
    Pour j=1 :length(Front{i}) - 1
        z = Front{i}([j,j+1]) ;
        k = 0 ;
        %p : variable d'arret du "tant que"
        p = 0 ;
        Tant que p==0
            %On cherche le numero global de l'arete de
            % sommets z ou fliplr(z) dans Am( :,1 :2) :
```

4. DESCRIPTION DU CODE

FONCTION deg_lib/maillage_triangle_arete_ps

```

    k = k + 1 ;
    a1 = sum(abs(z - Am(k,1 :2))) ;
    a2 = sum(abs(fliplr(z) - Am(k,1 :2))) ;
    Si a1==0 | a2==0
        %k est le numero globale de l'arete z
        Li = k ;
        %p : arret du "tant que" en posant p = 1
        p = 1 ;
        %Numero du triangle d'appartenance, numero local
        % d'arete sur le triangle et numero global d'arete
        arb(j, :) = [Pa(Li,1), Pa(Li,2), Li] ;
    Fin
Fin
Fin
Pfc{i} = arb ;
Fin
Pf se déduit alors de Pfc et Pa( :,5) de A et de Pf par :
Pf = [] ;
s = size(Pfc{i}) ;
Pour i=1 :length(Pfc)
    s = size(Pfc{i}) ;
    Pf = [Pf ; Pfc{i}, i*ones(s(1), 1)] ;
Fin
Pa = [Pa, zeros(size(A,1),1)] ;
Pa(Pf( :,3),5) = Pf( :,4) ;

```

Fonction :

```
function [Td, Ad, Xd, Yd] = maillage_ddl(rep, T, Am, Pa, X, Y) ;
```

Calcul des degrés de liberté (*ddl*). Les variables Td, Ad, Xd, Yd sont en particulier destinées à être stockées dans le fichier relatif au types d'espaces d'EF et au choix des formules d'intégration du Tableau 2.2.3-1b du Paragraphe 2.2.3 : les valeurs de Td, Ad, Xd, Yd relatives aux *ddl* du *i*-ème type d'espace choisis pour composer l'espace EF produit *E* sont respectivement stockées dans chaque cellule *i* des tableaux respectifs, de type CELL ARRAY, Tc, Arc, Xc et Yc.

Les *ddl* sont ceux décrits au Paragraphe 1.4. L'ordre de numérotation des *ddl* est celui donné au Paragraphe 2.3.3.

Dans le cas des espaces d'EF continus, l'ordre de numérotation choisi est celui des sommets du maillage, puis, dans l'ordre de parcours des arêtes de Am, des *ddl* sur les arêtes (hormis les sommets), puis dans l'ordre des triangles donnés par les lignes successives de T, des *ddl* intérieurs aux triangles.

- Sur chaque arête numéro *i* la numérotation se fait en allant du sommet Am(*i*, 1) vers le sommet Am(*i*, 2).

- La numérotation des *ddl* intérieurs aux triangles se fait de la manière suivante. Si a_1, a_2, a_3 sont les numéros de sommets du triangle *k* données dans leur ordre de numérotation locale sur ce triangle, *i.e.* $T(k, 1 : 3) = (a_1, a_2, a_3)$, on numérote alors les *ddl* intérieurs du triangle *i*, qui sont placés sur des droites parallèles à a_1a_2 , dans le sens a_1 vers a_2 et en partant de la droite la plus proche du segment a_1a_2 jusqu'à la droite la plus éloignée.

On reporte dans Td(*k*, :) les numéros de *ddl* portés par le triangle numéro *k* dans l'ordre suivant, d'abord les numéros de sommets du maillages T(*k*, 1 : 3), puis les numéros des *ddl* sur les arêtes : si a_1, a_2, a_3 sont des numéros de sommets du triangle *k* tels que $T(k, 1 : 3) = (a_1, a_2, a_3)$, alors les numéros des *ddl* d'arêtes sont reportés dans l'ordre suivant : d'abord ceux situés sur l'arête a_1a_2 en allant de a_1 vers a_2 (arête 1, localement), puis ceux de l'arête a_1a_3 en allant de a_1 vers a_3 (arête 2) puis ceux de l'arête a_2a_3 (arête 3), on termine enfin en reportant les numéros de *ddl* intérieurs au triangle suivant la règle énoncée plus haut pour ce type de *ddl*. Il est à noter que pour deux triangles *k* et *k'* ayant une arête commune, l'ordre de description dans Td(*k*, :) des *ddl* situés sur cette arête est soit l'ordre de ces *ddl* dans Td(*k'*, :) soit son inverse.

Dans le cas des espaces d'EF discontinus, l'ordre de numérotation, se fait triangle par triangle avec d'abord, les sommets du triangle, puis les *ddl* sur les arêtes (de a_1 vers a_2 , de a_1 vers a_3 , puis de a_2 vers a_3), puis les *ddl* intérieurs suivant le même principe que celui décrit précédemment.

Dans le cas du P_0 discontinu, les *ddl* sont les centres de gravité des triangles et chaque *ddl* a le même numéro que le triangle auquel il appartient.

Variables d'entrée :

Elles sont composées de variables de maillage **T**, **X** et **Y** du Tableau 2.2.2-1, Paragraphe 2.2.2, des variables **Am** et **Pa** calculées par `maillage_triangle_arete_ps` et de variable **rep** qui désigne le choix de l'espace d'EF.

rep : DOUBLE ARRAY. Numéro de l'espace d'EF choisi (*cf.* Paragraphe 1.4).

Pour $1 \leq i \leq 9$:

- **rep** = *i* désigne le P_i^c ,
- **rep** = 10*i* désigne le P_i^d (exemple, si *i* = 1, 10*i* désigne le nombre 101),

Pour $1 \leq i \leq 9$ et $0 \leq j \leq 9$:

- **rep** = *ij* désigne le $P_{i,j}^c$ défini en (1.21) au Paragraphe 1.4,
- **rep** = 1*ij* désigne le $P_{i,j}^d$ défini en (1.22) au Paragraphe 1.4.

Le P_0^d est désigné par **rep** = 100, le P_{10}^c par **rep** = 200 et le P_{10}^d par **rep** = 300.

La description de ces espaces d'EF est donnée au Paragraphe 1.4.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

X, **Y** : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Am : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes du maillage) $\times$ 2. Description arête par arête des sommets du maillage : **Am**(*i*, :) désigne les numéros des deux sommets de l'arête numéro *i* du maillage. L'arête n'est pas orientée. On a la relation **Am**(*i*, :) = **T**(**Pa**(*i*, 1), *z*) avec *z* = [1, 2] si **Pa**(*j*, 2) = 1, *z* = [1, 3] si **Pa**(*j*, 2) = 2, et *z* = [2, 3] si **Pa**(*j*, 2) = 3.

Pa : DOUBLE ARRAY. Matrice de taille (Nombre d'arête du maillage) $\times$ 5. Description des arêtes du maillage tel que **Pa**(*j*, 1) est un numéro de triangle auquel appartient l'arête *j* et **Pa**(*j*, 2) son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour **Pa**(*j*, 3 : 4) si l'arête appartient à un deuxième triangle sinon **Pa**(*j*, 3) et **Pa**(*j*, 4) sont nuls (arête sur la frontière). Enfin **Pa**(*j*, 5) est le numéro du morceau de frontière si l'arête *j* est sur la frontière, nul sinon ou si on a posé **Front** = []. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Front : CELL ARRAY. Vecteur colonne. La cellule *i* de **Front** est un vecteur ligne de type DOUBLE ARRAY qui décrit consécutivement sur la frontière les numéros de sommets du morceau de frontière numéro *i*, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

Td : DOUBLE ARRAY. Matrice de taille (Nombre de triangles) $\times$ (Nombre de *ddl* par éléments). Tableau décrivant les *ddl* triangles par triangles. **Td**(*i*, :) est le vecteur des numéros de *ddl* de l'EF désigné par **rep**. Dans le cas du P_0 discontinu, les *ddl* sont les centres de gravité des triangles et chaque *ddl* a le même numéro que le triangle auquel il appartient : **Td**(*i*, :) = *i*. Sinon, **Td**(*i*, 1 : 3) est le vecteur des

numéro des trois sommets du triangle i donné par $T(i, 1 : 3)$. Ensuite $Td(i, 4 : \text{end})$ indique les numéros des *ddl* situés sur les arêtes (hormis les sommets), puis à l'intérieur du triangle numéro i . Si a_1, a_2, a_3 sont les numéros de sommet d'un triangle i donnés dans leur ordre de numérotation locale sur ce triangle, $Td(i, 4 : \text{end})$ donne alors les numéros des *ddl* qui se succèdent dans l'ordre suivant : d'abord ceux de l'arête a_1a_2 en allant de a_1 vers a_2 (arête 1, localement), puis ceux de a_1a_3 en allant de a_1 vers a_3 (arête 2) puis ceux de a_2a_3 (arête 3) en allant de a_2 vers a_3 , puis les *ddl* intérieurs, ce en parcourant les droites parallèles à a_1a_2 , dans le sens a_1 vers a_2 et en partant de la droite la plus proche du segment a_1a_2 jusqu'à la plus éloignée.

Ad : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes) $\times$ (Nombre de *ddl* sur les arêtes, sommets compris) avec une seule colonne (nulle) s'il n'y pas de *ddl* sur les arêtes. Description des *ddl* arête par arête. Le terme $Ad(i, :)$ vaut 0 s'il n'y a pas de *ddl* d'arêtes comme pour les espaces d'EF discontinus. Sinon, $Ad(i, [1, 2])$ prend les valeurs des sommets de l'arête i (ces valeurs sont données par $Am(i, [1, 2])$, où **Am** est la variable de description des arêtes des sommets du maillage calculée par la fonction `maillage_triangle_arete_ps`). Ensuite, s'il y a des *ddl* sur cette arête privée de ses extrémités, $Ad(i, [3, \dots])$ prend les valeurs des numéros de ces *ddl* en les décrivant consécutivement du sommet d'arête $Ad(i, 1)$ vers le sommet d'arête $Ad(i, 2)$. Cette variable ne sert que dans la fonction `cb` pour les conditions de type Dirichlet où l'ordre de numérotation des *ddl* n'est pas utilisé.

Xd, Yd : DOUBLE ARRAY. Vecteur ligne de taille égale au nombre de *ddl*. Coordonnées en x et y des *ddl*.

Exemple : Cet exemple est donné dans le `help` de la fonction `maillage_ddl`, il reprend le maillage de deux triangles la figure 4.3-2 utilisée précédemment. Cet exemple met en œuvre le calcul des *ddl* du P_5^c .

EXEMPLE : Calcul des DDL du P_5 sur un maillage de deux triangles

```
T=[1,2,3 ;1,3,4] ;
X=[0,0,1,1] ;
Y=[1,0,0,1] ;
clear('Front')
Front{1}=[1,2,3,4,1] ;
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, Y, Front) ;
[Td, Ad, Xd, Yd] = maillage_ddl(5, T, Am, Pa, X, Y) ;
hold on ;
patch(X(T(1, :)),Y(T(1, :)), 'w') ;
patch(X(T(2, :)),Y(T(2, :)), 'w') ;
plot(Xd,Yd, 'rd') ;
```

La figure 4.3-3 donne le maillage, la numérotation des triangles, la numérotation globale des arêtes et les numéros des *ddl*.

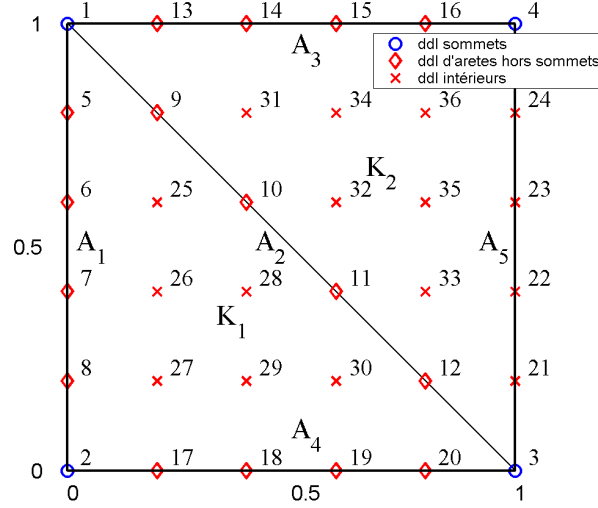


Figure 4.3-3. Représentation des *ddl* du P_5 , K_1 , K_2 sont les deux triangles du maillage, et $A_1, \dots, A_5$ les 5 arêtes du maillage dans l'ordre global des arêtes.

La valeur de $\mathbf{Ad}$ obtenue est :

$$\mathbf{Ad} = \begin{pmatrix} 1 & 2 & 5 & 6 & 7 & 8 \\ 1 & 3 & 9 & 10 & 11 & 12 \\ 1 & 4 & 13 & 14 & 15 & 16 \\ 2 & 3 & 17 & 18 & 19 & 20 \\ 3 & 4 & 21 & 22 & 23 & 24 \end{pmatrix},$$

et la valeur de $\mathbf{Td}$:

$$\mathbf{Td} = \begin{pmatrix} 1 & 2 & 3 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 17 & 18 & 19 & 20 & 25 & 26 & 27 & 28 & 29 & 30 \\ 1 & 3 & 4 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 21 & 22 & 23 & 24 & 31 & 32 & 33 & 34 & 35 & 36 \end{pmatrix}.$$

Si on reprend cet exemple avec la valeur de $\mathbf{T}$ suivante :

$$\mathbf{T} = [1, 2, 3 ; 3, 1, 4] ;$$

on remarque, sur la seconde ligne de la nouvelle matrice $\mathbf{Ad}$, l'inversion des 2 numéros de sommets 1 et 3 :

$$\mathbf{Ad} = \begin{pmatrix} 1 & 2 & 5 & 6 & 7 & 8 \\ 3 & 1 & 9 & 10 & 11 & 12 \\ 1 & 4 & 13 & 14 & 15 & 16 \\ 2 & 3 & 17 & 18 & 19 & 20 \\ 3 & 4 & 21 & 22 & 23 & 24 \end{pmatrix},$$

ce qui conduit à une numérotation inverse des *ddl* intérieurs 9, 10, 11, 12 de l'arête A_2 , comme représenté à la figure 4.3-4 ci-après. Ensuite, pour la nouvelle matrice

Td, on remarque l'inversion de l'ordre de ces *ddl* numéros 9, 10, 11, 12 sur Td(1, :). Cette inversion n'ayant pas lieu Td(2, :) car les sommets 1 et 3 ont aussi été inversé dans T(2, [1, 2]).

$$Td = \begin{pmatrix} 1 & 2 & 3 & 5 & 6 & 7 & 8 & 12 & 11 & 10 & 9 & 17 & 18 & 19 & 20 & 25 & 26 & 27 & 28 & 29 & 30 \\ 3 & 1 & 4 & 9 & 10 & 11 & 12 & 21 & 22 & 23 & 24 & 13 & 14 & 15 & 16 & 31 & 32 & 33 & 34 & 35 & 36 \end{pmatrix}.$$

La nouvelle figure correspondant à cette valeur de T est représentée ci-dessous :

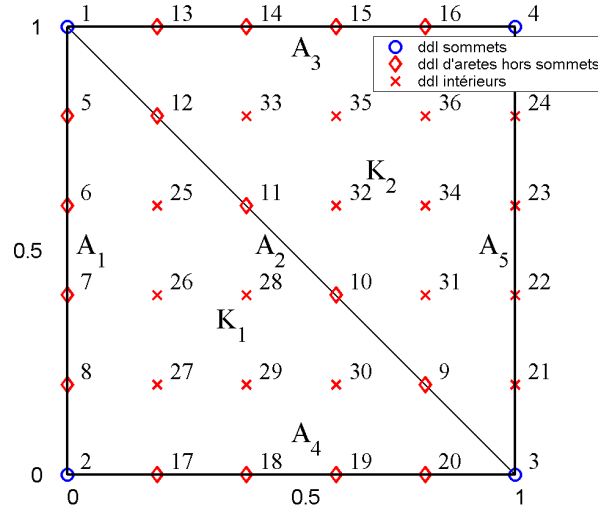


Figure 4.3-4. Pour $T = [1, 2, 3; 3, 1, 4]$, Représentation des *ddl* du P_5 , K_1 , K_2 sont les deux triangles du maillage, et $A_1, \dots, A_5$ les 5 arêtes du maillage dans l'ordre global des arêtes.

Avant de donner l'algorithme de la fonction, nous donnons deux boucles dont le principe est utilisé dans l'algorithme. Ces deux boucles permettent de décrire les deux ensembles suivants du Paragraphe 1.4 :

- l'ensemble $L_a^{int}(K)$ des *ddl* de l'espace vectoriel $P_k(K)$ qui appartiennent aux arêtes du triangle K ,
- l'ensemble $L_k^{int}(K)$ des *ddl* de l'espace vectoriel $P_k(K)$ qui sont à l'intérieur de K ,

avec une description qui suit l'ordre choisi au Paragraphe 1.4.

Notons a_1, a_2, a_3 les trois sommets du triangle K . On a

$$L_k^{int}(K) = \{a_{pqr} = \frac{p}{k}a_1 + \frac{q}{k}a_2 + \frac{r}{k}a_3, (p, q, r) \in \mathbb{N}^3, p+q+r = k, \min(p, q, r) = 0\}.$$

Cet ensemble se décrit en posant d'abord $r = 0$, avec q allant de 0 à k avec $p = k - q$ ce qui revient à décrire les *ddl* de l'arête de sommets a_1, a_2 en allant de a_1 vers a_2 . Puis en posant $q = 0$ avec r allant de 0 à k avec $p = k - r$ ce qui revient à décrire les *ddl* de l'arête de sommets a_1, a_3 en allant de a_1 vers a_3 et enfin en posant $p = 0$

avec r allant de 0 à k et $q = k - r$ ce qui revient à décrire les *ddl* de l'arête de sommets a_2, a_3 en allant de a_2 vers a_3 . Ce qui donne l'algorithme suivant.

On pose $r = 0$ puis,

Pour $q = 0$ à k :

$$\frac{p}{k}a_1 + \frac{q}{k}a_2, \text{ avec } p = k - q. \quad (4.1a)$$

Fin.

On pose $q = 0$ puis :

Pour $r = 0$ à k :

$$\frac{p}{k}a_1 + \frac{r}{k}a_3, \text{ avec } p = k - r. \quad (4.1b)$$

Fin.

On pose $p = 0$ puis :

Pour $r = 0$ à k :

$$\frac{q}{k}a_2 + \frac{r}{k}a_3, \text{ avec } q = k - r. \quad (4.1c)$$

Fin.

Pour les *ddl* intérieurs on a

$$L_k^{int}(K) = \{a_{pqr} = \frac{p}{k}a_1 + \frac{q}{k}a_2 + \frac{r}{k}a_3, (p, q, r) \in \mathbb{N}^3, p+q+r = k, \min(p, q, r) > 0\}.$$

Cet ensemble se décrit en faisant varier r de 1 à $k - 2$, ce qui revient à parcourir successivement les droites parallèles à la droite passant par a_1 et a_2 de troisième coordonnée barycentrique relativement à (a_1, a_2, a_3) donnée respectivement par $1/k, 2/k, \dots, (k - 2)/k$.

Ensuite, à r fixé, on fait varier q de 1 à $k - r - 1$ ce qui revient à décrire les *ddl* de la droite parallèle courante en allant de a_1 vers a_2 .

On obtient une description algorithmique de cet ordre d'énumération :

Pour $r = 1$ à $k - 2$:

Pour $q = 1$ à $k - r - 1$:

$$\frac{p}{k}a_1 + \frac{q}{k}a_2 + \frac{r}{k}a_3, \text{ avec } p = k - q - r. \quad (4.2)$$

Fin ;

Fin.

Algorithme :

EF/deg_lib/maillage_ddl.m

On remet ici la partie déclarative de la fonction :

```
function [Td, Ad, Xd, Yd] = maillage_ddl(rep, T, Am, Pa, X, Y) ;
```

Détermination en fonction de `rep` du choix de l'espace d'EF : $P_{i,j}^c$ (continu) ou $P_{i,j}^d$ (discontinu), le choix continu/discontinu ainsi que le choix i et j étant déterminé par les valeurs de `rep`.

Suivant la valeur de `rep` :

Traitement des espaces d'EF discontinus :

- Si `rep` = 100, traitement du P_0 discontinu.

Si `rep==100`

%Les degres de liberte sont les barycentres

Td = [1 :size(T,1)]' ;

%Barycentres

Xd = sum(X(T(:,1 :3)),2)/3 ;

Yd = sum(Y(T(:,1 :3)),2)/3 ;

Xd = Xd' ;

Yd = Yd' ;

%Degres de liberte sur les aretes : il n'y a pas de DDL sur

% sur les aretes

Ad = zeros(size(Am,1),1) ;

Fin ;

- Si `rep` $\in \{101, 102, \dots, 199, 300\}$, traitement du cas du $P_{i,j}$ discontinu avec

$i = j = 1$ si `rep`=101, pour le $P_{1,1}^d = P_1$ -discontinu usuel,

$i = j = 2$ si `rep`=102, pour le $P_{2,2}^d = P_2$ -discontinu usuel,

...

$i = j = 9$ si `rep`=109, pour le $P_{9,9}^d = P_9$ -discontinu usuel,

$i = j = 10$ si `rep`=300, pour le $P_{10,10}^d = P_{10}$ -discontinu usuel.

- Si `rep` $\in \{110, 111, \dots, 199\}$, traitement du cas du $P_{i,j}$ discontinu où i est le second chiffre de `rep` et j le troisième chiffre de `rep` :

- Par exemple `rep` = 134 donne $i = 3$, $j = 4$ qui correspond au $P_{3,4}^d$, espace P_3^d privé de ses fonctions «bulles» plus (somme directe) les bulles du P_4^d .

Suivant la valeur de `cas` au deux points ci-dessus, les variables `deg1` et `deg2` prennent les valeurs i et j dans $P_{i,j}^d$:

- `deg1` = i ;

- `deg2` = j ;

et `dimPK` prend la valeur de $\dim(P_{i,j}(K))$, $P_{i,j}(K)$ étant l'espace vectoriel défini au Paragraphe 1.4 :

Si `deg2==0%Pk classique`

`dimPK = 3*deg1 ;`

Sinon

`dimPK = 3*deg1 + (deg2 - 2)*(deg2 - 1)/2 ;`

Fin ;

• Boucle sur les triangles, on numérote les sommets, puis les arêtes, puis les *ddl* intérieurs dans l'ordre indiqué dans la présentation de la fonction ci-dessus.

Ci-dessous, pour $i = 1$, la variable `z` et ensuite `Td(i,1:dimPK)` prennent la valeur $(1, \dots, \text{dimPK})$, ensuite pour $i = 2$, ces deux termes prennent la valeur $(\text{dimPK} + 1, \dots, 2 * \text{dimPK})$ etc. Le vecteur `Td(i,1:dimPK)` désigne les numéros des *ddl* sur le triangle i .

Pour `i=1 :size(T,1) ;`

`z = dimPK*i - [dimPK-1 :-1 :0] ;`

`%Degres de libertes sur le triangles`

`Td(i,1 :dimPK) = z ;`

`%Coordonnees des DDL sommets`

`Xd(z(1 :3)) = [X(T(i,1 :3))]` ;

`Yd(z(1 :3)) = [Y(T(i,1 :3))]` ;

Calcul des coordonnées des *ddl* placés sur les arêtes. Posons $d = \text{deg1}$. Notons a_1, a_2, a_3 les sommets de numéros $T(i, 1)$, $T(i, 2)$ et $T(i, 3)$. Sans compter les *ddl* placés sur les sommets, il y a $d - 1$ *ddl* sur chaque arête.

En raisonnant de manière analogue à (4.1a, b, c) : les *ddl* de l'arête numéro 1 du triangle courant privé de ses sommets sont dans l'ordre de numérotation choisi pour les *ddl* :

$$\frac{k}{d}a_1 + \frac{d-k}{d}a_2, \text{ avec } k = d-1, d-2, \dots, 1,$$

sur l'arête numéro 2 du triangle courant ces *ddl* sont :

$$\frac{k}{d}a_1 + \frac{d-k}{d}a_3, \text{ avec } k = d-1, d-2, \dots, 1,$$

et sur l'arête numéro 3 du triangle courant ces *ddl* sont :

$$\frac{k}{d}a_2 + \frac{d-k}{d}a_3, \text{ avec } k = d-1, d-2, \dots, 1.$$

Les numéros de ces *ddl* sont donnés par `z(4 : 3*deg1)`. C'est le calcul des coordonnées qui ordonne les *ddl*.

Ci-dessous les valeurs de k/d et de $(d - k)/d$ avec $k = d - 1, d - 2, \dots, 1$ sont représentés vectoriellement par **k1** et **k2**.

```

k1 = [deg1-1 :-1 :1]/deg1 ;
k2 = 1 - k1 ;
Xd(z(4 : 4 + deg1 - 2)) = k1.*X(T(i,1)) + k2.*X(T(i,2)) ;
Xd(z(3 + deg1 : 2*deg1 + 1)) = ...
    k1.*X(T(i,1)) + k2.*X(T(i,3)) ;
Xd(z(2*deg1 + 2 : 3*deg1)) = ...
    k1.*X(T(i,2)) + k2.*X(T(i,3)) ;
Yd(z(4 : 4 + deg1 - 2)) = k1.*Y(T(i,1)) + k2.*Y(T(i,2)) ;
Yd(z(3 + deg1 : 2*deg1 + 1)) = ...
    k1.*Y(T(i,1)) + k2.*Y(T(i,3)) ;
Yd(z(2*deg1 + 2 : 3*deg1)) = ...
    k1.*Y(T(i,2)) + k2.*Y(T(i,3)) ;

```

Calcul des coordonnées des *ddl* situés à l'intérieur du triangle. Les numéros de ces *ddl* sont donnés par `z(3*deg1 : end)` à la suite des *ddl* sur les arêtes hors sommets. Posons $d = \text{deg2}$. On a vu en (4.2) que l'ordre de ces *ddl* était le suivant (en changeant le nom des variables) :

Pour $j = 1$ à $d - 2$:

Pour $i = 1$ à $d - j - 1$:

$$\frac{k_1}{d}a_1 + \frac{k_2}{d}a_2 + \frac{k_3}{d}a_3, \text{ avec } k_3 = j, k_2 = i, k_1 = d - k_3 - k_2.$$

Fin ;

Fin ;

Dans l'algorithme ci-dessous, on fait varier j de 1 à $d - 2$ et on pose

$$\mathbf{p1} = (1, 2, \dots, d - j - 1)$$

pour la boucle en i ci-dessus.

Les variables **k1**, **k2** et **k3** représentent sous forme vectorielles les valeurs de

$$\frac{k_1}{d}, \frac{k_2}{d}, \frac{k_3}{d}$$

avec :

$$\mathbf{k3} = \left(\frac{j}{d}, \frac{j}{d}, \dots, \frac{j}{d}\right), \mathbf{k2} = \left(\frac{1}{d}, \dots, \frac{d-j-1}{d}\right), \mathbf{k1} = \left(\frac{d-j-1}{d}, \dots, \frac{1}{d}\right).$$

Ci-dessous le calcul de `k3` se fait par

$$\mathbf{k3} = 1 - (\mathbf{k1} + \mathbf{k2}) = \left(\frac{j}{d}, \frac{j}{d}, \dots, \frac{j}{d}\right).$$

On donne la suite de l'algorithme, la variable `compt` permet de se positionner dans `Xd` et `Yd` :

```

    compt=3*deg1 ;
    Pour j=1 :deg2 - 2
        p1 = 1 :deg2 - j - 1 ;
Par exemple pour deg2 = 4 on aura p1 = [1,2], puis [1] ce qui donnera pour les
variables ci-dessous :
deg2 * k1 = [2, 1], puis [1],
deg2 * k2 = [1, 2], puis [1],
deg2 * k3 = [1, 1], puis [2],
Autre exemple, pour deg2 = 6 on aura p1 = [1,2,3,4], puis [1,2,3], puis [1,2],
puis [1], qui donnera pour les variables ci-dessous
deg2 * k1 = [4, 3, 2, 1], puis [3, 2, 1], puis [2, 1], puis [1],
deg2 * k2 = [1, 2, 3, 4], puis [1, 2, 3], puis [1, 2], puis [1],
deg2 * k3 = [1, 1, 1, 1], puis [2, 2, 2], puis [3, 3], puis [4].
    k1 = fliplr(p1)/deg2 ;
    k2 = p1/deg2 ;
    %Par construction k3 vaut j/deg2 sur chacune de ses
    % composante, sa longueur est celle de k1 et k2 :
    k3 = 1 - (k1 + k2) ;
    Xd(z(compt + 1 :compt + length(p1))) = ...
        k1.*X(T(i,1)) + k2.*X(T(i,2)) + k3.*X(T(i,3)) ;
    Yd(z(compt + 1 :compt + length(p1))) = ...
        k1.*Y(T(i,1)) + k2.*Y(T(i,2)) + k3.*Y(T(i,3)) ;
    compt = compt + length(p1) ;
Fin ;
Fin;%de la boucle sur les triangles
• Fin de la boucle sur les triangles,
    - Pour les conditions au bord on considère que les ddl sont situés à l'intérieur
    des triangles et on pose alors pour  $i = 1, \dots$ , nombre d'arêtes :
    Ad(i,1) = 0 ;

```

- Fin du traitement du cas du $P_{i,j}$ discontinu.

Traitement des espaces d'EF continus :

- Si $\text{rep} \in \{1, 2, \dots, 99\}$, traitement du cas du $P_{i,j}$ continu avec :
 - $i = j = 1$ si $\text{rep}=1$, pour le $P_{1,1}^c = P_1$ -continu usuel,
 - $i = j = 2$ si $\text{rep}=2$, pour le $P_{2,2}^c = P_2$ -continu usuel,
 - ...
 - $i = j = 9$ si $\text{rep}=9$, pour le $P_{9,9}^d = P_9$ -continu usuel,
 - $i = j = 10$ si $\text{rep}=200$, pour le $P_{10,10}^d = P_{10}$ -continu usuel.

- Si $\text{rep} \in \{10, 11, \dots, 99\}$, i est le premier chiffre de rep et j le second chiffre de rep . Par exemple $\text{rep} = 134$ donne $i = 3$, $j = 4$ qui correspond au $P_{3,4}^c$, espace P_3^c privé de ses fonctions «bulles» plus (somme directe) les bulles du P_4^c .

Ci-dessous les variables `deg1` et `deg2` désignent les valeurs de i et j dans $P_{i,j}^c$. On pose

- `deg1` = i ;

- `deg2` = j ;

et

Si `deg2==0`%Pk classique

`dimPK` = $3 \times \text{deg1}$;

Sinon

`dimPK` = $3 \times \text{deg1} + (\text{deg2} - 2) \times (\text{deg2} - 1) / 2$;

Fin ;

Si `deg1` <= 1, on place dans `Td` les *ddl* sommets du maillage :

`Td` = `T` ;

On place ensuite dans `Ad` les *ddl* qui sont les sommets de chaque arête du maillage :

`Ad` = `Am` ;

et on passe au *ddl* intérieurs car il n'y pas de *ddl* d'arêtes.

Si `deg1` > 1, on maille les arêtes successives décrites par `Am` de la manière suivante. Tout d'abord on attribue des numéros aux arêtes intérieures allant de 1 au nombre d'arêtes $\mathcal{N}_A$ multiplié par `deg1` - 1 :

`A` = `reshape(1 : size(Am,1)*(deg1 - 1), deg1 - 1, size(Am,1))'` ;

Exemple pour le P_4 , le $P_{4,0}$, le $P_{4,1}$, ..., le $P_{4,9}$ on obtient en posant $d = \text{deg1} = 4$:

$$A = \begin{pmatrix} 1 & 2 & 3 = d - 1 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ \dots & \dots & \dots \\ \mathcal{N}_A(d - 1) - 2 & \mathcal{N}_A(d - 1) - 1 & \mathcal{N}_A(d - 1) \end{pmatrix}$$

et pour d quelconque :

$$A = \begin{pmatrix} 1 & 2 & \dots & d-1 \\ 4 & 5 & \dots & 2(d-1) \\ \dots & \dots & \dots & \dots \\ \mathcal{N}_A(d-1) - d + 2 & \mathcal{N}_A(d-1) - d + 1 & \dots & \mathcal{N}_A(d-1) \end{pmatrix}.$$

On obtient alors Ad par concaténation de Am qui décrit les sommets d'arêtes, arêtes par arêtes et de la matrice $\text{Noeu} + A$ obtenue en ajoutant à la matrice A la valeur du nombre de sommets du maillage :

`%Degres de libertes sur les aretes :`

`%Noeu ci-dessous est le nombre de sommets du maillage`

`Ad = [Am, Noeu + A] ;`

• Boucle sur les arêtes (effectuée si `deg1 > 1`). On place les numéros de *ddl* dans le tableau Td de description par triangles des *ddl* par une boucle sur les arêtes. Cette construction est faite avec une matrice intermédiaire C telle que $\text{C} + \text{Noeu}$ est la partie de Td concernant ces *ddl*.

Pour chaque arête numéro i on considère le triangle numéro $\text{Pa}(i, 1)$ et l'éventuel triangle numéro $\text{Pa}(i, 3)$.

`Pour i=1 :size(Am,1)`

- Cas du premier triangle. Les *ddl* de l'arête i de sommets numéros $\text{Am}(i, :)$, et qui ont pour numéro $\text{A}(i, :)$, sont donc sur l'arête numéro $\text{Pa}(i, 2)$ (nombre valant 1, 2 ou 3) du triangle numéro $\text{Pa}(i, 1)$. Les sommets de $\text{Am}(i, :)$ ont par construction le même ordre que ceux de l'arête $\text{Pa}(i, 2)$ du triangle numéro $\text{Pa}(i, 1)$. On pose donc pour le triangle numéro $\text{Pa}(i, 1)$:

`C(Pa(i,1), [(deg1-1)*Pa(i,2)-deg1+2 : (deg1-1)*Pa(i,2)]) ...
= A(i, :) ;`

sachant qu'après la boucle on posera :

`Td = [T, C + Noeu] ;`

- Cas d'un deuxième triangle. Si $\text{Pa}(i, 3) > 0$, l'arête est commune à un deuxième triangles. Les *ddl* de l'arête i , de sommets de numéro $\text{Am}(i, :)$ et qui ont pour numéro $\text{A}(i, :)$ sont donc sur l'arête numéro $\text{Pa}(i, 4)$ du triangle numéro $\text{Pa}(3, 1)$. Les sommets de $\text{Am}(i, :)$ peuvent avoir l'ordre inverse que ceux de l'arête $\text{Pa}(i, 4)$ de ce triangle, dans ce cas l'ordre des *ddl* doit être inverse :

`%Cas d'un arete commune a un deuxieme triangle`

`Si Pa(i,3)>0`

`v1 = Am(i, :) ;`

`%r prend les valeur 1 ou 2 ou 3 :`

`r = Pa(i,4) ;`


```

%z vaut [1,2] si r = 1, [1,3] si r = 2 et [2,3] si r = 3
z = (r-3)*(r-2)*[1,2]/2 + (3-r)*(r-1)*[1,3] ...
    + (r-1)*(r-2)*[2,3]/2 ;
v2 = T( Pa(i,3),z ) ;
Si v1==v2 %(ordre des sommets de l'arete identique)
    C(Pa(i,3),[(deg1-1)*Pa(i,4)-deg1+2 :(deg1-1)*Pa(i,4)]) ...
    = A(i, :) ;
sinon %(ordre des sommets de l'arete inverse)
    C(Pa(i,3),[(deg1-1)*Pa(i,4)-deg1+2 :(deg1-1)*Pa(i,4)]) ...
    = fliplr(A(i, :)) ;
Fin ;

Fin ;

Fin ; %de la boucle sur les aretes
• Fin de la boucle sur les arêtes. On pose
Td = [T,C + Noeu] ;
• Coordonnées des degrés de liberté placés sur les arêtes. On numérote en allant
des sommets Am( :,1) vers les sommets Am( :,2).
Xd = zeros(1,length(A( :))) ;
Yd = Xd ;
k = (deg1-1 :-1 :1)./deg1 ;
Pour p=1 :deg1-1
    Xd(A( :,p)) = k(p)*X(Am( :,1)) + k(deg1 - p)*X(Am( :,2)) ;
    Yd(A( :,p)) = k(p)*Y(Am( :,1)) + k(deg1 - p)*Y(Am( :,2)) ;
Fin ;
Xd = [X,Xd] ;
Yd = [Y,Yd] ;
ce qui termine le cas deg1 > 1 pour la numérotation des arêtes.
• Boucle sur les triangles pour la numérotation des ddl intérieurs, on traite aussi
le cas du  $P_{1,j}$  non concerné par les ddl d'arêtes précédents. Posons  $d = \text{deg2}$ . On
utilise une variante de la boucle du cas discontinu. On numérote les ddl intérieurs
des triangles successifs décrits par Td (ou T) avec des numéros allant de 1 au
nombre total  $\mathcal{N}_T$  de triangles multiplié par  $d = (d-2)(d-1)/2$  qui est le nombre
de ddl intérieurs :
Si deg2>=3
    I = [] ;

```

```

%Nombre de DDL interieurs
n_int = (deg2 - 2)*(deg2 - 1)/2 ;
%Nombre de DDL deja numerotes
M = max(max(Td)) ;
%Numerotation des DDL interieurs aux triangles
B = reshape(1 :size(Td,1)*n_int, n_int, size(Td,1))' ;

```

Exemple pour le P_4 , le $P_{1,4}$, le $P_{2,4}$, ..., le $P_{9,4}$ on obtient en posant $d' = (\deg2 - 2) * (\deg2 - 1)/2$:

$$B = \begin{pmatrix} 1 & 2 & \dots & d' \\ 4 & 5 & \dots & 2d' \\ \dots & \dots & \dots & \dots \\ \mathcal{N}_T d' - d' + 1 & \mathcal{N}_T d' - d' + 2 & \dots & \mathcal{N}_T d' \end{pmatrix}$$

```

%Construction finale de Td :
Td = [Td, M + B] ;
%Les degres de liberte sont les n_int sommets interieurs

```

- Début de la boucle sur les triangles.

```

Pour i=1 :size(Td,1)
    %on a aussi z = B(i, :) :
    z = [n_int*i - n_int + 1 :n_int*i] ;
    %Coordonnees des sommets du triangle courant :
    Xp1 = X(T(i,1)) ;
    Xp2 = X(T(i,2)) ;
    Xp3 = X(T(i,3)) ;
    %
    Yp1 = Y(T(i,1)) ;
    Yp2 = Y(T(i,2)) ;
    Yp3 = Y(T(i,3)) ;

```

Boucle sur les *ddl* intérieurs. Notons a_1, a_2, a_3 les sommets de numéros $T(i, 1)$, $T(i, 2)$ et $T(i, 3)$. Comme vu en (4.2) les *ddl* intérieurs sont, en changeant au besoin les noms des variables, donnés par la boucle :

Pour $k = 1$ à $d - 2$:

Pour $j = 1$ à $d - k - 1$:

$$\frac{i}{d}a_1 + \frac{j}{d}a_2 + \frac{k}{d}a_3, \text{ avec } i = d - j - k.$$

Fin ;

Fin,

ou, de manière équivalente :

Pour $k = 1$ à $d - 2$:

Pour $i = d - k - 1$ à 1 :

$$\frac{i}{d}a_1 + \frac{j}{d}a_2 + \frac{k}{d}a_3, \text{ avec } j = d - k - i.$$

Fin ;

Fin.

On reprend ci-après la suite de l'algorithme, basée sur cette boucle :

```
%Description des noeuds interieurs
compt = 0 ;
z1 = [] ;
z2 = [] ;
z3 = [] ;
Pour k=1 :deg2 - 2,
    Pour i=deg2 - k - 1 :-1 :1,
        j=deg2 - k - i ;
        compt = compt + 1 ;
        z1(compt) = i/deg2 ;
        z2(compt) = j/deg2 ;
        z3(compt) = k/deg2 ;
    Fin ;
Fin ;
Xd(M + z) = z1*Xp1 + z2*Xp2 + z3*Xp3 ;
Yd(M + z) = z1*Yp1 + z2*Yp2 + z3*Yp3 ;
Fin ;
```

Fin ; %de la boucle sur les triangles

- Fin du traitement du cas du $P_{i,j}$ continu et fin de la fonction.

Fonctions de EF/mat_cell.**Fonction :**

```
function C = add_mat_cell(A, B) ;
```

Addition de matrices de type `CELL` par blocs. Cette fonction additionne des matrices type `CELL ARRAY` dont les cellules sont des matrices des type `DOUBLE ARRAY` ou `SPARSE ARRAY`.

Pour `A`, `B` de type `CELL ARRAY` de tailles identiques, tels que $A\{i, j\}$ et $B\{i, j\}$ soient des matrices de type `DOUBLE ARRAY` ou bien `SPARSE ARRAY` et de tailles identiques, alors

$$C = \text{ADD\_MAT\_CELL}(A, B)$$

est un tableau de type `CELL ARRAY` de même taille que `A` et `B` et tel que

$$C\{i, j\} = A\{i, j\} + B\{i, j\}.$$

Si les deux cellules $A\{i, j\}$ et $B\{i, j\}$ sont de type `SPARSE ARRAY` alors $C\{i, j\}$ est aussi de type `SPARSE ARRAY`. Sinon $C\{i, j\}$ est de type `DOUBLE ARRAY`.

Variables d'entrée : `A`, `B` : `CELL ARRAY`. Matrices à additionner par blocs.

Variables de sortie :

`C` : `CELL ARRAY`. Résultat de l'addition par blocs.

Exemple : Cet exemple est donné dans le `help` de la fonction `add_mat_cell`.

EXEMPLE

```
A = {[1,2],[3,4] ; [5,6],[7,8]} ;
B = {[8,7],[6,5] ; [4,3],[2,1]} ;
C = add_mat_cell(A,B) ;
disp(A{1,1}) ;
disp(B{1,1}) ;
disp(C{1,1}) ;
```

Algorithme :

EF/mat_cell/add_mat_cell.m

On remet ici la partie déclarative de la fonction :

```
function C = add_mat_cell(A, B) ;
```

L'algorithme s'écrit :

```
Pour i=1 :size(A,1)
```

```
    Pour j=1 :size(A,2)
```

```
        C{i,j} = A{i,j} + B{i,j} ;
```

```
    Fin ;
```

```
Fin ;
```

Fonction :

```
function Uc = init_transf_cell(rep, U, donnees, varargin) ;
```

Initialisation d'un vecteur colonne de type `CELL ARRAY` représentant les composantes d'une fonction de l'espace EF produit courant E ou bien conversion d'un vecteur de type `DOUBLE` en vecteur de type `CELL ARRAY` pour passer d'une représentation à une autre d'un élément de l'espace E (cf. Paragraphe 1.5).

La commande

```
Uc = init_transf_cell('init', [], donnees) ;
```

permet l'initialisation de `Uc` au vecteur nul de l'espace EF produit E décrit par le champ `donnees.D.f` de la variable `donnees` et le fichier `.mat` associé au champ `donnees.nom_prep` de `donnees` pour l'utilisation de la variable `Ed` (cf. Tableau 2.2.3-1b du Paragraphe 2.2.3).

Le vecteur `Uc` représente une fonction de l'espace EF produit $E = E_1 \times \dots \times E_n$ dans la représentation décrite au Paragraphe 1.5.

La commande «transparente» :

```
Uc = init_transf_cell('init', [], donnees, X, Y, T) ;
```

réalise la même opération à partir des champs `donnees.D.E` et `donnees.D.f` et des variables de maillage `X`, `Y`, `T`.

La commande

```
Uc = init_transf_cell('ta2c', U, donnees) ;
```

permet en particulier de convertir le vecteur `U` issu de la résolution de $SU = F$ où `U` et `F` sont issus de la phase d'assemblage :

```
[S,F] = assemblage_EF(donnees) ;
```

en tenant compte du fait que `U` est une représentation d'un sous-espace de E ou de E privé de certaines de ses composantes (cf. (1.2) et (1.3)').

Cette fonction initialise `Uc` à un vecteur colonne de type `CELL ARRAY` tel que le contenu des cellules i donné par `Uc{i}` est un vecteur colonnes de type `DOUBLE ARRAY`. Donnons des exemples d'utilisation courante de cette fonction.

Utilisation courante.

A - Initialisation au vecteur nul de l'espace EF produit E décrit par `donnees.D` :

1) En utilisant les variables de maillage :

```
Uc = init_transf_cell('init', [], donnees, X, Y, T) ;
```

Dans ce cas `donnees` doit avoir les champs suivants initialisés :

- `donnees.D.f`,
- `donnees.D.E`,

et `X`, `Y`, `T` sont les variables définissant le maillage.

2) Situation la plus courante pour initialiser, avec un fichier de données créé en général par la fonction `EF_Prep`, la fonction nulle de l'espace EF produit courant E :

```
Uc = init_transf_cell('init', [], donnees);
```

Dans ce cas `donnees` doit avoir les champs suivants initialisés :

- `donnees.nom_prep` et le fichier désigné par cette variable doit contenir la variable `Ed` donnant la dimension des espaces d'EF composant l'espace EF produit E (cf. Tableau 2.2.3-1b du Paragraphe 2.2.3).

- `donnees.D.f`.

En sortie `Uc` est un vecteur colonne de type `CELL ARRAY` dont les cellules sont des vecteurs colonnes de type `SPARSE ARRAY`.

Voici deux exemples d'utilisation de la fonction sur ces deux cas, ces exemples sont donnés dans le `help` de la fonction `init_transf_cell`.

EXEMPLE A-1

```
T = [1,2,3;1,3,4] ;
X = [0,0,1,1] ;
Y = [1,0,0,1] ;
clear donnees
donnees.D.E = [1,2,101,23] ;
%E = P1 x P2 x P1d x P2 + bulle P3 x P1d,
%   dim(E) = 4 x 9 x 6 x 11 x 6
donnees.D.f = [1,2,3,4,3] ;
Uc = init_transf_cell('init', [], donnees, X, Y, T);
disp(Uc);
```

EXEMPLE A-2, avec fichier de donnees

```
Ed = [4,9,6,11,6]
%Creation du fichier relatif aux EF
a = mkdir('C :/EF/mat_cell/temp');
if a==0, mkdir('C :/EF/mat_cell/temp'), end;
save('C :/EF/mat_cell/temp/exemple','Ed');
clear donnees
donnees.nom_prep = 'C :/EF/mat_cell/temp/exemple';
donnees.D.f = [1,2,3,4,3];
```

```
Uc = init_transf_cell('init', [], donnees);
disp(Uc);
```

B - Conversion d'un vecteur colonne de type `DOUBLE` ou `SPARSE ARRAY` en vecteur colonne de type `CELL`.

1) A partir du maillage :

```
Uc = init_transf_cell('ta2c', U, donnees, X, Y, T);
```

Dans ce cas `donnees` doit avoir les champs suivants initialisés :

```
donnees.D.f,
donnees.D.E,
```

et `X`, `Y`, `T` sont les variables définissant le maillage.

2) Situation la plus courante, en général après la résolution du système issu du problème EF et de solution le vecteur `U` :

```
Uc = init_transf_cell('ta2c', U, donnees);
```

Dans ce cas `donnees` doit avoir les champs suivants initialisés :

```
donnees.nom_prep et le fichier désigné doit contenir Ed,
donnees.D.f.
```

3) Avec duplication d'une variable ou ajout d'une composante, avec, suivant le cas initialisation de :

```
- donnees.D.inc_point (supression de cellules),
```

ou de

```
- donnees.D.inc_cell (supression de composantes).
```

Par exemple si

- `donnees.D.inc_cell = [i; j]` (vecteur colonne), ceci signifie que `U` représente alors au départ, les composantes d'une fonction de E privée de sa j -ème composante, ce qui correspond à une élimination d'inconnue système.

Ensuite, si

- `donnees.D.inc_point = {i, j, k}`; i, j entiers et k nombres réel,

`U` représente alors les composantes d'une fonction privée de la composante j de la i -ème composante de cette fonction sur E .

En sortie la variable `Uc` est un vecteur colonne de type `CELL ARRAY` dont les cellules sont des vecteurs colonnes du même type, `SPARSE` ou `DOUBLE ARRAY`, que la variable `U`.

Voici trois exemples d'utilisation de la fonction sur ces cas, ces exemples sont donnés dans le `help` de la fonction `init_transf_cell`.

Dans les deux exemples B-1 et B-2, `Uc` prend les valeurs de 1 à 36 :

EXEMPLE B-1

```

T = [1,2,3;1,3,4] ;
X = [0,0,1,1] ;
Y = [1,0,0,1] ;
clear donnees
donnees.D.E = [1,2,101,23] ;
%E = P1 x P2 x P1d x P2 + bulle P3 x P1d,
%   dim(E) = 4 x 9 x 6 x 11 x 6
donnees.D.f = [1,2,3,4,3] ;
U = [1 :36]' ;
Uc = init_transf_cell('ta2c', U, donnees, X, Y, T) ;
for i=1 :5, disp(Uc{i}'), end ;

```

EXEMPLE B-2

```

Ed = [4,9,6,11,6]
%Creation du fichier relatif aux EF
a = mkdir('C :/EF/mat_cell/temp') ;
if a==0, mkdir('C :/EF/mat_cell/temp'), end ;
save('C :/EF/mat_cell/temp/exemple','Ed') ;
clear donnees
donnees.nom_prep = 'C :/EF/mat_cell/temp/exemple' ;
donnees.D.f = [1,2,3,4,3] ;
U = [1 :36]' ;
Uc = init_transf_cell('ta2c', U, donnees) ;
for i=1 :5, disp(Uc{i}'), end ;

```

L'exemple B-3 utilise une variable `donnees.D.inc_cell` qui est un vecteur colonne de 2 termes et une variable `donnees.D.inc_point` qui est un vecteur de 3 cellules. On peut généraliser en utilisant pour `donnees.D.inc_cell` une matrice de deux lignes et pour `donnees.D.inc_point` une matrice de type `CELL` de 3 colonnes, comme décrit après cet exemple, ainsi que dans l'exemple 1 ci-après (fichier *exemple1-init_transf_cell.m*).

EXEMPLE B-3

```

T = [1,2,3;1,3,4] ;
X = [0,0,1,1] ;
Y = [1,0,0,1] ;
clear donnees

```



```

donnees.D.E = [1,2,101,23] ;
%E = P1 x P2 x P1d x P2 + bulle P3 x P1d,
%   dim(E) = 4 x 9 x 6 x 11 x 6
donnees.D.f = [1,2,3,4,3] ;
disp('Duplication de la composante 3 mise en 5') ;
U = [1 :30]' ;
donnees.D.inc_cell = [3;5] ;
Uc = init_transf_cell('ta2c', U, donnees, X, Y, T) ;
for i=1 :5, disp(Uc{i}'), end ;
donnees.D.inc_cell = [] ;
disp('Insertion de 0 en position 2 de la 3-ieme composante') ;
U = [1 :35]' ;
donnees.D.inc_point = {3,2,0} ;
Uc = init_transf_cell('ta2c', U, donnees, X, Y, T) ;
for i=1 :5, disp(Uc{i}'), end ;

```

Utilisation générale.

La variable `donnees` est la variable de description des données, décrite au Paragraphe 2.2.

La variable `U` est vide ou de type `DOUBLE ARRAY`, ou `SPARSE ARRAY` et sert à initialiser la variable de sortie `Uc` comme décrit ci-après.

En sortie, la variable `Uc` est de longueur $n = \text{length}(\text{donnees.D.f})$ qui est le nombre termes de l'espace produit E . Le champ `donneesD.f`, décrit au Tableau 2.2.1-1c du Paragraphe 2.2.1, permet de définir l'espace EF produit E à partir des types d'espaces d'EF choisis.

Ensuite, le vecteur `Uc{i}` est un vecteur colonne de type `DOUBLE ARRAY` ou `SPARSE ARRAY` de longueur égale à $\text{dim}(E_i)$ donnée par le terme `Ed(donnees.D.f(i))`. La variable `Ed` est celle du Tableau 2.2.3-1b du Paragraphe 2.2.3, elle s'obtient, à l'intérieur de la fonction `init_transf_cell` soit par la commande

```
- load(donnees.nom_prep,'Ed')
```

si l'on n'utilise pas de variables optionnelles (3 variables d'entrée), soit par la commande

```
- [Xc, Yc, Tc, Ed] = ddl(donnees,X,Y,T,[]),
```

si on utilise les 3 variables optionnelles de maillage et `X`, `Y`, `T` (6 variables d'entrée), la variable `donnees` est appelée pour la valeur de son champ `donnees.D.E` qui définit les types d'espaces d'EF utilisés (*cf.* Tableau 2.2.1-1c du Paragraphe 2.2.1).

Pour `rep = 'init'` et $U = []$ ou quelconque, le contenu de $Uc\{i\}$ est le vecteur colonne nul de taille $Ed(donnees.D.f(i))$ et Uc représente, suivant la représentation du Paragraphe 1.5 des fonctions de E , la fonction nulle de E .

Pour `rep = 'ta2c'`, la fonction transforme le vecteur colonne U en vecteur de type `CELL ARRAY` Uc . Si les champs `inc_point` et `inc_cell` de `donnees.D` ne sont pas définis ou sont vides (cas fréquent) on aura alors

$$U = [Uc\{1\}, \dots, Uc\{n\}].$$

On peut définir cette fonction comme la fonction réciproque de la fonction suivante.

Soit Uc de type `CELL ARRAY` tels que $Uc\{i\}$ est un vecteur colonne de type `DOUBLE ARRAY` de longueur $Ed(donnees.D.f(i))$.

Pour chaque valeur de i entre 1 et le nombre de lignes du tableau de cellules `donnees.D.inc_point`, on suppose que les composantes données par le vecteur `donnees.D.inc_point{i,2}` du vecteur $Uc\{donnees.D.inc_point\{i,1\}\}$ sont données par les valeurs du vecteur `donnees.D.inc_point{i,3}`.

Ensuite on suppose que pour chaque valeur de i allant de 1 au nombre de ligne de la matrice donnée par `donnees.D.inc_cell` on a l'égalité

$$Uc\{donnees.D.inc_cell(2,i)\} = Uc\{donnees.D.inc_cell(1,i)\}$$

excepté pour les composantes correspondant à `donnees.D.inc_point`.

Alors U s'obtient en éliminant dans Uc les composantes indiquées comme ci-dessus par les deux premières colonnes de `donnees.D.inc_point` et en supprimant les cellules de Uc dont les numéros sont donné par la deuxième ligne de la matrice `donnees.D.inc_cell`, puis en posant

$$U = [Uc\{1\}; \dots; Uc\{end\}].$$

Pour cette valeur de U la fonction `init_transf_cell` permet alors de retrouver Uc . En général dans les problèmes EF, on partira de la définition de Uc en fonction de la formulation EF choisie.

Variables d'entrée :

`rep` : `CHAR ARRAY`. Prend soit la valeur `'init'` pour l'initialisation de Uc au vecteur nul de E , soit `'ta2c'` pour une initialisation en fonction d'un vecteur U ci-après.

U : `DOUBLE ARRAY` ou `SPARSE ARRAY`. Soit $U = []$, soit U est un vecteur colonne. Dans le code EF la variable U sera du type de la solution du système assemblé par `EF_assemblage(donnees)`.

`donnees` : `STRUCT ARRAY`. Variable décrivant les données du problème EF.

Champs de `donnees` à initialiser. Pour la commande

```
Uc = init_transf_cell(rep, [], donnees, X, Y, T);
```

on doit initialiser

`donnees.D.f` (Tableau 2.2.1-1-c, Paragraphe 2.2.1)

`donnees.D.E` (Tableau 2.2.1-1-c, Paragraphe 2.2.1)

et `X`, `Y`, `T` sont les variables définissant le maillage (*cf.* Paragraphe 2.2.2).

Pour la commande

```
Uc = init_transf_cell(rep, [], donnees);
```

Les champs suivants de `donnees` doivent être initialisés : `donnees.nom_prep` dont le fichier associé doit contenir `Ed`, et `donnees.D.f`.

Ensuite, si `rep = 'ta2c'`, en fonction de `U` et des éliminations d'inconnues, les champs suivants sont initialisés :

- `donnees.D.inc_point` : [] ou matrice de `CELL` de 3 colonnes,
- `donnees.D.inc_cell` : [] ou matrice de `DOUBLE` de deux lignes.

`donnees.nom_prep` : `CHAR ARRAY`. Nom du fichier relatif au espaces d'EF et aux formules d'intégration (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1). Ici ce fichier sert pour charger la valeur de `Ed`.

`donnees.D.inc_point` : `CELL ARRAY`. Ensemble vide ou tableau de 3 colonnes. Les termes de la première colonne sont des scalaires indiquant les cellules sur lesquelles on reconstitue certaines valeurs de `Uc`. Les termes de la seconde colonne sont des vecteurs lignes ou colonnes, ou le terme '`end`', et dont chaque terme désigne les lignes à reconstituer. Les termes de la troisième colonne sont des vecteurs lignes ou colonnes qui donnent les valeurs à attribuer aux lignes à reconstituer.

`donnees.D.inc_cell` : `DOUBLE ARRAY`. Ensemble vide ou matrice. Les termes de la deuxième ligne indiquent les inconnues cellules à reconstituer. Si une cellule est reconstituée à partir d'une autre cellule, le nombre de composantes consacrées à chacune de ces deux cellules dans `donnees.D.inc_point` doit être identique. Les termes de la seconde ligne de `donnees.D.inc_cell` doivent être tous distincts et les 2 lignes de `donnees.D.inc_cell` doivent être deux ensembles disjoints.

Les variables de maillages sont optionnelles et sont utilisées à la place du fichier `donnees.nom_prep`, le champ `donnees.D.E` doit alors être initialisé (*cf.* Tableau 2.2.1-1-c, Paragraphe 2.2.1).

`varargin{1}`, `varargin{2}`, `varargin{3}`. Termes `X` et `Y` et `T` de maillage vus dans les appels de fonction ci-dessus :

`X,Y` : `DOUBLE ARRAY`. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

`Uc` : `CELL ARRAY`. Vecteur colonne de taille `length(donnees.D.f)`. Les cellules de `Uc{i}` sont des vecteurs colonnes de taille `Ed(donnees.D.f(i))`, `Ed` étant la variable du fichier relatif aux espaces d'EF aux formules d'intégration qui donne la dimension de chaque type d'espace d'EF choisi (*cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3). Dans le cas `U = []` et `rep = 'init'`, le vecteur `Uc{i}` est de type `SPARSE ARRAY`, sinon ce vecteur est de même type que `U` : `DOUBLE ARRAY` ou `SPARSE ARRAY`. Dans le code EF, `Uc` est une représentation des fonctions de l'espace EF produit E (*cf.* Paragraphe 1.5).

Exemple 1 : Cet exemple, indiqué dans le `help` de la fonction `init_transf_cell`, est donné par la commande :

EXEMPLE 1

```
cd C :/EF/mat_cell, exemple1_init_transf_cell
```

Dans cet exemple l'espace E a pour dimension $5 \times 5 \times 4 \times 5 \times 5 \times 5$.

Au 1) de cet exemple le vecteur `Uc1` est initialisé au vecteur nul de E . Il vaut

$$\mathbf{Uc1} = \{[0; 0; 0; 0; 0]; [0; 0; 0; 0; 0]; [0; 0; 0; 0; 0]; [0; 0; 0; 0; 0]; [0; 0; 0; 0; 0]\}.$$

Au 2) de cet exemple le vecteur `Uc2` est calculé à partir du vecteur

$$\mathbf{U} = [4; 7; 5; 7; 8; 8; 10; 11].$$

Comme on a `donnees.D.f = (2, 2, 1, 2, 2)` de longueur 5, `Uc2` aura donc 5 cellules. Ensuite, la variable `donnees.D.inc_cell` prend la valeur

$$\text{donnees.D.inc\_cell} = \begin{pmatrix} 5 & 5 \\ 1 & 2 \end{pmatrix},$$

elle indique que les cellules 1 et 2 de `Uc2` seront recopiées à partir de `Uc2{5}` après que les valeurs de `U` soient distribuées sur les 3 autres cellules restantes.

Enfin, la variable `donnees.D.inc_point` de type `CELL ARRAY` prend la valeur :

```

donnees.D.inc_point =
{  1,   2,  2000;
  1, 'end', 1000;
  2,   2,   100;
  2, 'end', 2000;
  4,   2,   100;
  4, 'end', 2000;
  5,   2,   100;
  5, 'end', 2000;
  3,   2,    10;
  3,   3,  1000}
```

elle indique ici en particulier que chaque cellule comporte l'insertion de deux termes.

Dans la fonction `init_transf_cell`, la variable `Uc2` prend alors les valeurs successives suivantes. Suivant, `donnees.D.inc_cell`, on distribue les valeurs de `U` sur les 5 cellules, hormis les cellules 1 et 2, et le vecteur `Uc2` vaut alors :

$$Uc2 = \{[]; []; [4; 7]; [5; 7; 8]; [8; 10; 11]\},$$

puis, par duplication de la cellule 5 :

$$Uc2 = \{[8; 10; 11]; [8; 10; 11]; [4; 7]; [5; 7; 8]; [8; 10; 11]\}.$$

Enfin, comme indiqué par la variable `donnees.D.inc_point`, on obtient finalement pour `Uc2`, par insertion de composantes :

$$Uc2 = \{[8; 2000; 10; 11; 1000]; [8; 100; 10; 11; 2000]; \\ [4; 10; 1000; 7]; [5; 100; 7; 8; 2000]; [8; 100; 10; 11; 2000]\}.$$

La fin de l'exemple effectue l'opération réciproque qui consiste à retrouver `U` à partir de `U2` et à mettre le résultat un tableau de cellules `Uc3`. Pour cela on utilise la fonction `supp_lin_col_vec_cell` qui élimine sur `Uc2` les composantes indiquées par `donnees.D.inc_point`, ce qui donne

$$Uc3 = \{[8; 10; 11]; [8; 10; 11]; [4; 7]; [5; 7; 8]; [8; 10; 11]\}.$$

Enfin, par élimination des 2 premières cellules de `Uc3`, on retrouve `U` à partir de `Uc2`, sous forme du tableau de cellules `Uc3` :

$$Uc3 = \{[4; 7]; [5; 7; 8]; [8; 10; 11]\}.$$

L'algorithme de cet exemple est le suivant :

C :/EF/mat_cell/exemple1_init_transf_cell.m

```
%Exemple pour init_transf_cell
%Commande
%   cd C :/EF/mat_cell, exemple1_init_transf_cell
%
%Dans l'exemple 1 ci-dessous on pose Ed = [4,5] et
%
%   donnees.D.f = [2,2,1,2,2]
%
%   qui correspondent aussi a un espace EF
```

```

%
%   E = E1 x E2 x E3 x E4 x E5 compose de deux type
%
%   d'espaces F1 F2, de dimension 4 et 5 tels que
%
%   E3 = F1, E1 = E2 = E4 = E5 = F2.
%
%Dans 2) ci-dessous, la taille de U est donnee par
%
%   sum(Ed(donnees.D.f)) moins
%   la taille des cellules 1 et 2 qui sont supprimees moins le
%   nbre de termes de donnees D.inc_point hors ceux des cellules
%   1 et 2,
%
%ce qui donne ici 24 - 10 - 6 = 8 composantes.

clear
Ed = [4,5] ;
%Creation du fichier relatif aux EF
a = mkdir('C :/EF/mat_cell/temp') ;
if a==0, mkdir('C :/EF/mat_cell/temp'), end ;
save('C :/EF/mat_cell/temp/exemple','Ed')
donnees.nom_prep = 'C :/EF/mat_cell/temp/exemple' ;
donnees.D.f = [2,2,1,2,2] ;
%
%1) Initialisation de Uc1 a 0 :
Uc1 = init_transf_cell('init',[],donnees) ;
disp('1) Uc1, vecteur nul de l''espace EF courant,') ;
disp('ligne par ligne :') ;
for i=1 :5, disp(full(Uc1{i})), end ;

%2) Construction d'un Uc a partir d'un U
%Uc{1}(2) vaudra 2000, Uc{1}(5) vaudra 1000, etc.
donnees.D.inc_point = {

```

```

1, 2, 2000 ;
1, 'end', 1000 ;
2, 2, 100 ;
2, 'end', 2000 ;
4, 2, 100 ;
4, 'end', 2000 ;
5, 2, 100 ;
5, 'end', 2000 ;
3, 2, 10 ;
3, 3, 1000};
%Uc{1} = Uc{2} = Uc{5} :
donnees.D.inc_cell = [5, 5 ;
                      1, 2] ;

%U doit etre de taille 8 :
U = [4, 7, 5, 7 ,8, 8, 10, 11]';
format compact;
disp('2) Valeur de U'', Ed(donnees.D.f), donnees.D.inc_cell');
disp('et de donnees.D.inc_point :');
disp(U');
disp(Ed(donnees.D.f));
disp(donnees.D.inc_cell);
disp(donnees.D.inc_point);
disp('Uc{i} = [] pour i=1,2 = ligne 2 de donnees.D.inc_cell,');
disp('initialisation de Uc2{i}, i=3,4,5, complementaire de [1,2]');
disp('dans [1,2,3,4,5]), avec U : [Uc2{3}; Uc2{4}; Uc2{5}] = U,');
disp('puis initialisation de Uc2{1} = Uc2{2} qui valent Uc2{5},');
disp('puis insertion des composantes mentionnees ');
disp('par donnees.D.inc_point le tout avec init_transf_cell')
disp('et affichage des valeurs de Uc2{i}''', i=1,...,5 :');
Uc2 = init_transf_cell('ta2c', U, donnees);
for i=1 :5, disp(Uc2{i}'), end
disp('Reciproque : calcul de U a partir de Uc2 par');
disp('Uc3 = supp_lin_col_vec_cell(Uc2, donnees.D)');

```

```

disp('et Uc3([1,2]) = [] : ');
Uc3 = supp_lin_col_vec_cell(Uc2, donnees.D);
Uc3([1,2]) = [];
disp('on retrouve Uc3 qui vaut U,');
disp('valeur de [Uc3{1}; Uc3{2}; Uc3{3}]'' :');
disp([Uc3{1}; Uc3{2}; Uc3{3}]');
format ;

```

Exemple 2 : Cet exemple est donné dans le `help` de la fonction `init_transf_cell`. La fonction `init_transf_cell` est utilisée dans l'exemple sur le Laplacien du Paragraphe 2.1.4 pour initialiser l'itéré courant à 0, l'exemple est donné par la commande :

EXEMPLE 2

```
cd C :/EF_text/exemples/exemple1_Laplacien, instruct1
```

L'algorithme de la fonction `init_transf_cell` est le suivant.

Algorithme :

`EF/mat_cell/init_transf_cell.m`

On remet ici la partie déclarative de la fonction :

```
function Uc = init_transf_cell(rep, U, donnees, varargin);
```

La variable `Ed` est chargée par

```
load(donnees.nom_prep, 'Ed');
```

ou, si `varargin` est vide, calculée par

```
%Utilise donnees.D.E
```

```
[Xc, Yc, Tc, Ed] = ddl(donnees,X,Y,T,[]);
```

Ensuite

```
D = donnees.D;
```

```
%Dimension des espaces produits :
```

```
Dnc = Ed(D.f);
```

```
Si rep=='init'
```

```
Uc = cell(length(D.f),1);
```

```
Pour i=1 :length(D.f);
```

```
Uc{i,1} = sparse(Dnc(i),1);
```

```
Fin;
```

```
Fin;
```

```
Si rep=='ta2c'
```

```
Uc = trans_sol_mat_sol_cell(U,D,Dnc);
```

```
Fin
```


Fonction :

```
function Uc = trans_sol_mat_sol_cell(U, D, Dnc) ;
```

Transformation d'un vecteur colonne `U`, de type `DOUBLE ARRAY` ou `SPARSE ARRAY`, en un vecteur colonne `U` de type `CELL ARRAY`. La transformation est effectuée en fonction de la variable `Dnc` qui est un vecteur ligne d'entiers dont la longueur détermine le nombre de cellules de `Uc` et le terme `Dnc(i)` le nombre d'éléments de `Uc{i}` et en fonction des champs `D.inc_point` et `D.inc_cell` qui déterminent avec `U` le contenu des cellules de `Uc`.

En sortie `Uc` est un vecteur de type `CELL ARRAY` dont les cellules contiennent un vecteur colonne de même type que `U` : `DOUBLE ARRAY` ou `SPARSE ARRAY`.

Dans le cadre du code EF `U` et `Uc` représentent les composantes d'une fonction de l'espace EF produit courant E (cf. Paragraphe 1.5). Le terme `Dnc` est tel que `Dnc(i)` est la dimension du i -ème espace d'EF de l'espace EF produit, sa longueur est donc égal à n le nombre de termes E_i dans E .

La commande

```
Uc = trans_sol_mat_sol_cell(U, D, Dnc) ;
```

initialise `Uc` à un vecteur colonne de type `CELL ARRAY`, de `length(Dnc)` cellules. La cellule `Uc{i}` est un vecteur colonne de type `DOUBLE ARRAY` ou `SPARSE ARRAY` de taille `Dnc(i)`.

1) Si `D.inc_point` et `D.inc_cell` sont vides ou si `D.inc_point` et `D.inc_cell` ne sont pas des champs de `D` (cas fréquent) alors `Uc` vérifie :

$$[Uc\{1\}; \dots; Uc\{n\}] = U$$

et donc `U` en entrée doit être de taille `sum(Dnc)`.

2) Si `D.inc_cell` est un vecteur comme par exemple `[2; 3]` alors `Uc` vérifiera dans ce cas :

$$[Uc\{1\}; Uc\{2\}; Uc\{4\}; \dots; Uc\{n\}] = U$$

et `U` en entrée devra être de taille `sum(Dnc) - Dnc(3)`.

3) Si `donnees.D.inc_point` est une tableau de cellules comme par exemple

$$D.inc\_point = \{n, Dnc(n), 0\}$$

alors `Uc` vérifiera

$$[Uc\{1\}; \dots; Uc\{n\}] = [U; 0]$$

et donc `U` en entrée devra être de taille `sum(Dnc) - 1`.

Autre exemple pour `D.inc_point = {1, 4, 0}` alors `Uc` vérifiera avec `length(U) = sum(Dnc) - 1` :

$$[Uc\{1\}; \dots; Uc\{n\}] = [U(1 : 3); 0; U(4 : 'end')].$$

Remarque. On pourra aussi poser `D.inc_point = {n, 'end', 0}` à la place de `D.inc_point = {n, Dnc(n), 0}` ce qui donnera le même résultat.

4) De manière générale `Uc` vérifie :

a) Valeur imposée sur la composante d'une cellule de `Uc` :

$$Uc\{D.inc\_point\{i, 1\}\}(D.inc\_point\{i, 2\}) = D.inc\_point\{i, 3\}.$$

b) Egalité de deux cellules de `Uc` :

$$Uc\{i\} = Uc\{j\} \text{ si } D.inc\_cell(2, i) = D.inc\_cell(1, j),$$

exception faite des composantes qui entrent dans le cas a).

c) Le vecteur `[Uc{1}; ...; Uc{n}]`, auquel on supprime les composantes

$$Uc\{D.inc\_point\{i, 1\}\}(D.inc\_point\{i, 2\}),$$

puis les cellules suivantes en posant

$$Uc(D.inc\_cell(2, :)) = []$$

doit redonner le contenu, dans l'ordre, de `U` (ce qui donne la taille d'entrée de `U`).

Le terme `D` est la variable de type `struct` Tableau 2.2.1-1-c, Paragraphe 2.2.1) dont seuls les champs `D.inc_point`, `D.inc_cell` sont utilisés dans la fonction `trans_sol_mat_sol_cell`.

On pourra obtenir la variable `Dnc`, qui est le vecteur des dimensions des espaces d'EF E_i composant l'espace EF produit courant E , soit par

```
load(donnees.nom_prep, 'Ed') ;
```

```
Dnc = Ed(donnees.D.f) ;
```

soit, en utilisant les variables variables de maillage `X`, `Y`, `T` et la valeur de `donnees.D.E`, par les commandes

```
[Xc, Yc, Tc, Ed] = ddl(donnees, X, Y, T, []);
```

et le terme `Dnc(i)` est alors la dimension de l'espace E_i .

La commande

```
Uc = trans_sol_mat_sol_cell(U, D, Dnc) ;
```

est équivalente à la commande

```
Uc = init_transf_cell('ta2c', U, donnees) ;
```

qui permet de reconstituer `Uc` à partir de `U`, avec `D = donnees.D` (Tableau 2.2.1-1-c, Paragraphe 2.2.1). En pratique c'est cette dernière commande qui sera utilisée.

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/trans_sol_mat_sol_cell`

Variables d'entrée :

U : `DOUBLE ARRAY` ou `SPARSE ARRAY`. **U** est un vecteur colonne. Dans le code EF, **U** est de type solution du système assemblé par `EF_assemblage(donnees)`.

D : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe 2.2.1.

Les champs de **D** utilisés pour passer de **U** à **Uc** sont `D.inc_point` et `D.inc_cell`.

`D.inc_point` : `CELL ARRAY`. Ensemble vide ou tableau de 3 colonnes. Les termes de la première colonne sont des scalaires indiquant les cellules sur lesquelles on reconstitue certaines valeurs de **Uc**. Les termes de la seconde colonne sont des vecteurs lignes ou colonnes, ou le terme 'end', et dont chaque terme désigne les lignes à reconstituer. Les termes de la troisième colonne sont des vecteurs lignes ou colonnes qui donnent les valeurs à attribuer aux lignes à reconstituer.

`D.inc_cell` : `DOUBLE ARRAY`. Ensemble vide ou matrice. Les termes de la deuxième ligne indiquent les inconnues cellules à reconstituer. Si une cellule est reconstituée à partir d'une autre cellule, le nombre de composantes consacrées à chacune de ces deux cellules dans `D.inc_point` doit être identique. Les termes de la seconde ligne de `D.inc_cell` doivent être tous distincts et les 2 lignes de `D.inc_cell` doivent être deux ensembles disjoints.

Les termes `D.inc_point` et `D.inc_cell` sont facultatifs, on peut aussi poser `D = []` dans cette fonction.

Dnc : `DOUBLE ARRAY`. Vecteur ligne d'entiers. Dans le code EF, `Dnc(i)` représentera la dimension du *i*-ème espace d'EF de l'espace EF produit *E*.

Variables de sortie :

Uc : `CELL ARRAY`. Vecteur colonne de taille `length(Dnc)`. Les cellules de `Uc{i}` sont des vecteurs colonnes de taille `Dnc(i)`. Les cellules `Uc{i}` sont du même type que **U**, soit `DOUBLE ARRAY`, soit `SPARSE ARRAY`. Dans le code EF, **Uc** est une représentation des fonctions de l'espace d'EF produit *E* et `Dnc(i)` donne la dimension du *i*-ème espace du produit (*cf.* Paragraphe 1.5).

Exemple : Cet exemple est donné dans le `help` de la fonction `trans_sol_mat_sol_cell`. Le calcul de **Uc2** ci-dessous est identique à celui vu et détaillé dans l'exemple 1 de la fonction précédente, avec l'utilisation de `trans_sol_mat_sol_cell` à la place de `init_transf_cell`.

EXEMPLE

```
%Dans cet exemple on pose Dnc = [5,5,4,5,5] ce qui correspond a
%l'espace EF E = E1 x E2 x E3 x E4 x E5. La taille de U est
%donnee par sum(Dnc) - (taille des cellules 1 et 2 qui sont
%supprimees) - (nbre termes de D.inc_point hormis ceux des
```

%cellules 1 et 2), ce qui donne $24 - 10 - 6 = 8$ composantes.

```
clear ;
Dnc = [5, 5, 4, 5, 5] ;
%Uc{1}(2) vaudra 2000,Uc{1}(5) vaudra 1000, etc. :
D.inc_point={
    1, 2, [2000] ;
    1, 'end', 1000 ;
    2, 2, 100 ;
    2, 'end', 2000 ;
    4, 2, 100 ;
    4, 'end', 2000 ;
    5, 2, 100 ;
    5, 'end', 2000 ;
    3, 2, 10 ;
    3, 3, 1000} ;
%Uc{1} = Uc{2} = Uc{5} :
D.inc_cell = [5,5 ;
              1,2] ;
%U doit etre de taille 8
U = [4, 7, 5, 7, 8, 8, 10, 11]' ;
Uc2 = trans_sol_mat_sol_cell(U, D, Dnc) ;
disp('On donne ci-dessous Uc2 cellule par cellule : ');
disp('On a Uc2{1} = Uc2{2} = Uc2{5}, avec D.inc_cell. ');
disp('On la relation U = [Uc2{3} ; Uc2{4} ; Uc2{5}] ');
disp('qui s''obtient en enlevant de Uc2 les composantes')
disp('mentionnees par D.inc_point et les cellules donnees')
disp('par D.inc_cell(2, :).')
for i=1 :5, disp(Uc2{i}'), end ;
```

Algorithme :

EF/mat_cell/trans_sol_mat_sol_cell.m

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/trans_sol_mat_sol_cell`

On remet ici la partie déclarative de la fonction :

```
function Uc = trans_sol_mat_sol_cell(U, D, Dnc) ;
```

On commente l'algorithme avec l'exemple précédent du `help` où on a posé :

$$U = [4; 7; 5; 7; 8; 8; 10; 11], \quad Dnc = [5, 5, 4, 5, 5]$$

et

$$D.inc\_cell = \begin{bmatrix} 5 & 5 \\ 1 & 2 \end{bmatrix}, \quad D.inc\_point : \quad \left\{ \begin{array}{lll} [1] & [2] & [2000] \\ [1] & 'end' & [1000] \\ [2] & [2] & [100] \\ [2] & 'end' & [2000] \\ [4] & [2] & [100] \\ [4] & 'end' & [2000] \\ [5] & [2] & [100] \\ [5] & 'end' & [2000] \\ [3] & [2] & [10] \\ [3] & [3] & [1000] \end{array} \right\}.$$

Dans l'exemple, le vecteur de cellules `Uc2` final issu de la transformation de `U` par la fonction `trans_sol_mat_sol_cell` aura alors 5 cellules de tailles respectives données par `Dnc = [5, 5, 4, 5, 5]`.

• 2 Etapes d'élimination.

- La première étape consiste à enlever de `Dnc` les numéros d'inconnues ponctuelles et les inconnues cellules. L'opération est effectuée sur `Dnc2` qui vaut `Dnc2 = Dnc` au départ.

Ensuite, la variable de sortie `Uc` finale s'obtient par ajout de cellules puis d'inconnues ponctuelles à un vecteur cellule initial noté ici `UcI` composé des éléments successifs de `U`. Réciproquement, on voit que l'on passe de `Uc` à `UcI` en posant `UcI = Uc` puis en enlevant à `UcI` les composantes mentionnées par `D.inc_point`, puis en posant `UcI{j} = []` pour $j \in D.inc\_cell(2, :)$.

Ainsi, dans l'exemple, `Uc2` a 5 cellules de tailles respectives données par `Dnc = [5, 5, 4, 5, 5]`, en enlevant les composantes mentionnées par `D.inc_point` on obtient un vecteur à 5 cellules de tailles respectives données par `[3, 3, 2, 3, 3]`, puis en enlevant les cellules mentionnées par la deuxième ligne de `D.inc_cell`, un vecteur à 3 cellules de tailles respectives données par `[2, 3, 3]` qui porte sur les cellules de numéros successifs donnés par `[3, 4, 5]` de `Uc2`.

- La seconde étape consiste alors à calculer les 2 vecteurs correspondant au deux termes `[2, 3, 3]` et `[3, 4, 5]` ci-dessus.

$$Dnc2 = Dnc ;$$

• Etape 1 : Elimination des inconnues ponctuelles : on parcourt les lignes de `D.inc_point`, à chaque ligne on diminue `Dnc2(r)` de `length(m)` termes où r est la cellule concernée et m les numéros d'inconnues concernées.

Pour `j=1 :size(D.inc_point,1)`

`r = D.inc_point{j,1};%Numero de cellule sur laquelle`

`%porte l'elimination`

`m = D.inc_point{j,2};%Numero de ligne de`

`%la cellule a eliminer`

`Si D.inc_point{j,2}=='end'`

`m = Dnc(r) ;`

`Fin ;`

`Dnc2(r) = Dnc2(r) - length(m) ;`

`Fin ;`

Dans l'exemple on ici `Dnc2 = [3,3,2,3,3]`.

• Etape 2 : Elimination des inconnues de type cellules : On pose $Dnc2(r) = []$, si r est sur la deuxième ligne de la matrice `D.inc_cell` :

`%A est la liste des inconnues a qui on retire les inconnues appar-`

`% tenant a D.inc_cell(2, :) et on pose Dnc2(D.inc_cell(2, :))=[] ;`

`A=1 :length(Dnc) ;`

`i = D.inc_cell ;`

`Si isempty(i)==0`

`Si any(Dnc(i(1, :))~=Dnc(i(2, :)))`

`error(['les variables cellules sont choisies ', ...`

`'identiques leur Dnc doit l''etre aussi'])`

`Fin ;`

`Dnc2(i(2, :))=[] ;`

`A=setdiff(A,i(2, :)) ;`

`Fin ;`

Ici, dans le cas de l'exemple, `A` vaut `[3,4,5]` qui désigne les numéros des cellules restantes et `Dnc2 = [2,3,3]` les tailles respectives de ces cellules.

`%cs sommes cumulees, repere les emplacement des blocs dans U :`

`cs = cumsum(Dnc2) ;`

`cs = [0,cs] ;`

• Remplissage des cellules de `Uc` de numéro appartenant à la liste `A` avec `U`. La cellule `A(i)` contenant les termes de `cs(i) + 1` à `cs(i + 1)` de `U`.

Dans l'exemple on a

`cs = [0,2,5,8],`

et on pose :

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/trans_sol_mat_sol_cell`

$$\begin{aligned} \text{Uc}\{A(1), 1\} &= \text{Uc}\{3\} = \text{U}(\text{cs}(1) + 1 : \text{cs}(2)) = [4; 7], \\ \text{Uc}\{A(2), 1\} &= \text{Uc}\{4\} = \text{U}(\text{cs}(2) + 1 : \text{cs}(3)) = [5; 7; 8], \\ \text{Uc}\{A(3), 1\} &= \text{Uc}\{5\} = \text{U}(\text{cs}(3) + 1 : \text{cs}(4)) = [8; 10; 11]. \end{aligned}$$

L'algorithme s'écrit :

```
Pour i=1 :length(Dnc2)
    Uc{A(i),1}=U(cs(i) + 1 : cs(i+1));
Fin ;
```

A cette étape, sur l'exemple, Uc vaut alors :

$$\text{Uc} = \{[]; []; [4; 7]; [5; 7; 8]; [8; 10; 11]\};$$

- Ajout des cellules identiques :

```
Si isfield(D, 'inc_cell')
    Si isnumeric(D.inc_cell)
        Pour i=D.inc_cell
            Uc{i(2),1}=Uc{i(1),1};
        Fin ;
    Sinon
        error(['Le champ D.inc_cell doit etre', ...
            'de type numeric array']);
    Fin ;
Fin ;
```

Ici Uc vaut alors :

$$\text{Uc} = \{[8; 10; 11]; [8; 10; 11]; [4; 7]; [5; 7; 8]; [8; 10; 11]\};$$

- Insertion dans Uc des termes provenant de `D.inc_point`. On commence par ordonner `D.inc_point`, l'ordre étant donné par la première colonne. Tout d'abord on met dans un vecteur cette première colonne.

%Valeur de la colonne 1 de `D.inc_point` :

```
Pour i=1 :size(D.inc_point,1)
    C1(i) = D.inc_point{i,1};
Fin ;
```

On obtient dans l'exemple :

$$\text{C1} = [1, 1, 2, 2, 4, 4, 5, 5, 3, 3].$$

```
%J donne l'ordre de D.inc_point :
[I,J] = sort(C1) ;
%D2.inc_point vaut D.inc_point a l'ordre pres :
%   la premiere colonne de D2.inc_point est ordonnee
Pour i=1 :size(D.inc_point,1)
    D2.inc_point{i,1} = D.inc_point{J(i),1} ;
    D2.inc_point{i,2} = D.inc_point{J(i),2} ;
    D2.inc_point{i,3} = D.inc_point{J(i),3} ;
Fin ;
On obtient dans l'exemple :
```

$$I = [1, 1, 2, 2, 3, 3, 4, 4, 5, 5]; \quad J = [1, 2, 3, 4, 9, 10, 5, 6, 7, 8]$$

et

$$D2.inc\_point : \begin{array}{ccc} \{ & [1] & [2] & [2000] \\ & [1] & 'end' & [1000] \\ & [2] & [2] & [100] \\ & [2] & 'end' & [2000] \\ & [3] & [2] & [10] \\ & [3] & [3] & [1000] \\ & [4] & [2] & [100] \\ & [4] & 'end' & [2000] \\ & [5] & [2] & [100] \\ & [5] & 'end' & [2000] \end{array} \}.$$

• On réunit toutes les références à une cellule en une seule référence : la première colonne de `D3.inc_point` est alors composée des nombres de 1 à `length(Dnc)` avec des cellules sur les colonnes 2 et 3 qui peuvent être vides (`[]`). Sur l'exemple cela donne :

$$D3.inc\_point : \begin{array}{ccc} \{ & [1] & [2 \times 1 \text{ double}] & [2 \times 1 \text{ double}] \\ & [2] & [2 \times 1 \text{ double}] & [2 \times 1 \text{ double}] \\ & [3] & [2 \times 1 \text{ double}] & [2 \times 1 \text{ double}] \\ & [4] & [2 \times 1 \text{ double}] & [2 \times 1 \text{ double}] \\ & [5] & [2 \times 1 \text{ double}] & [2 \times 1 \text{ double}] \end{array} \}$$

avec

$$\begin{aligned} D3.inc\_point\{1,2\} &= \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \quad D3.inc\_point\{1,3\} = \begin{pmatrix} 2000 \\ 1000 \end{pmatrix}, \\ D3.inc\_point\{2,2\} &= \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \quad D3.inc\_point\{1,3\} = \begin{pmatrix} 100 \\ 2000 \end{pmatrix}, \\ D3.inc\_point\{3,2\} &= \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \quad D3.inc\_point\{1,3\} = \begin{pmatrix} 10 \\ 1000 \end{pmatrix}, \end{aligned}$$

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/trans_sol_mat_sol_cell`

$$D3.inc\_point\{4,2\} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}, D3.inc\_point\{1,3\} = \begin{pmatrix} 100 \\ 2000 \end{pmatrix},$$

$$D3.inc\_point\{5,2\} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}, D3.inc\_point\{1,3\} = \begin{pmatrix} 100 \\ 2000 \end{pmatrix}.$$

Pour cela l'algorithme s'écrit :

```
Pour i=1 :length(Dnc)
    D3.inc_point{i,1} = i ;
    D3.inc_point{i,2} = [] ;
    D3.inc_point{i,3} = [] ;
Fin ;

%Pour i=1 :size(D2.inc_point,1), on reunit les D3.inc_point{M1,2}
% tels que M1 = D2.inc_point{i,1}, idem pour D3.inc_point{M1,3} :
Pour i=1 :size(D2.inc_point,1)
    M1 = D2.inc_point{i,1} ;
    M2 = D2.inc_point{i,2} ;
    M3 = D2.inc_point{i,3} ;
    Si ischar(M2)
        Si M2=='end'
            M2=Dnc(M1) ;
        Sinon
            error('Chaine de caractere doit etre egale a end')
        Fin ;
    Fin ;
    D3.inc_point{M1,2} = [D3.inc_point{M1,2}( :) ; M2( :)] ;
    D3.inc_point{M1,3} = [D3.inc_point{M1,3}( :) ; M3( :)] ;
Fin ;
```

- Insertion des termes et fin de la fonction. Par construction de `D3.inc_point` on ne passe qu'une seule fois sur une cellule de `Uc` et avec `m` ci-dessous, de manière ordonnée à l'intérieur de celle-ci.

```
%Insertion : par construction de D3.inc_point on ne passe
% qu'une seule fois sur une cellule (necessaire pour inserer)
Pour j=1 :size(D3.inc_point, 1)
    r = D3.inc_point{j,1} ; %Numero de cellule sur laquelle
        % porte l'insertion
    m = D3.inc_point{j,2} ; %Colonne de numeros de la ligne dans
```

4. DESCRIPTION DU CODE

FONCTION `mat_cell/trans_sol_mat_sol_cell`

```

                                %      la cellule a inserer
    val = D3.inc_point{j,3};%Colonne de valeurs de la ligne dans
                                %la cellule a inserer

    %Rajout des inconnues.
    %      On ordonne m sinon ca ne marche pas :
    [I,J] = sort(m) ;
    mJ = m(J) ;
    valJ = val(J) ;
    %Insertion :
    Pour i=1 :length(mJ)
        Uc{r} = [Uc{r}(1 :mJ(i)-1) ; valJ(i) ; Uc{r}(mJ(i) :end)] ;
    Fin ;
Fin ;
On obtient alors la valeur finale de Uc :

Uc = {[8; 2000; 10; 11; 1000]; [8; 100; 10; 11; 2000];
      [4; 10; 1000; 7]; [5; 100; 7; 8; 2000]; [8; 100; 10; 11; 2000]}.
```

Fonction :

```
function Sc = trans_mat_mat_cell(S, D, Dnc) ;
```

Transformation d'une matrice **S** de type `DOUBLE` ou `SPARSE ARRAY` en une matrice par blocs représentée par une variable de type `CELL ARRAY` dont le contenu des cellules est de type `DOUBLE ARRAY` ou `SPARSE ARRAY`. La variable **S** est soit une matrice carrée soit un vecteur ligne ou colonne.

Si **S** est une matrice carrée, l'instruction

```
Sc = trans_mat_mat_cell(S, D, Dnc) ;
```

donne une matrice **Sc** de type `CELL ARRAY` qui représente la décomposition par blocs de la matrice **S**, les blocs (i, j) étant de dimension $Dnc(i) \times Dnc(j)$. Le bloc (i, j) de **S** est donné par $Sc\{i, j\}$. La matrice par blocs représentée par **Sc** est carrée.

Si **U** est un vecteur

```
Uc = trans_mat_mat_cell(U, D, Dnc) ;
```

met **U** sous forme d'un vecteur par blocs tels que $Uc\{i\}$ est de dimension $Dnc(i)$. Si **U** est un vecteur ligne (resp. colonne), **Uc** est un vecteur ligne (resp. colonne) dont les cellules sont des vecteurs lignes (resp. colonnes).

La variable **D** n'est pas utilisée ici et peut être mise à `[]` dans cette fonction.

Dans l'utilisation de la fonction `trans_mat_mat_cell` dans la fonction d'assemblage `EF_assemblage` la matrice **S** sera de taille égale à la dimension de l'espace `EF` produit *E* et $Dnc(i)$ représentera la dimension du *i*-ème espace d'EF de l'espace `EF` produit *E*.

Variables d'entrée :

S : `DOUBLE ARRAY` ou `SPARSE ARRAY`. Matrice carrée à décomposer en matrice par blocs.

D : `STRUCT ARRAY`. Non utilisée. Peut être mise à `[]` dans cette fonction.

Dnc : `DOUBLE ARRAY`. Vecteur ligne. Délimite la taille des blocs.

Variables de sortie :

Sc : `CELL ARRAY`. Représentation par blocs de **S**, sous forme de tableau de cellules. Les blocs ont le même type que **S** : `DOUBLE ARRAY` ou `SPARSE ARRAY`.

Exemple : Cet exemple est donné dans le `help` de la fonction `trans_mat_mat_cell`.

EXEMPLE

```
Dnc = [2,3,2] ;
S = pascal(7) ;
```

```

U = [1 :7]' ;
Sc = trans_mat_mat_cell(S, [], Dnc) ;
Uc = trans_mat_mat_cell(U, [], Dnc) ;
disp('contenu de Sc : ')
[Sc{1,1},Sc{1,2},Sc{1,3} ;
 Sc{2,1},Sc{2,2},Sc{2,3} ;
 Sc{3,1},Sc{3,2},Sc{3,3}]
disp('contenu de Uc : ')
[Uc{1} ; Uc{2} ; Uc{3}]

```

Cet exemple décompose Uc en :

$$Uc = \{[1; 2]; [3; 4; 5]; [6; 7]\};$$

et Sc en :

$$Sc = \left\{ \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 1 \\ 3 & 4 & 5 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 6 & 7 \end{pmatrix} \right. \\ \begin{pmatrix} 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{pmatrix}, \begin{pmatrix} 6 & 10 & 15 \\ 10 & 20 & 35 \\ 15 & 35 & 70 \end{pmatrix}, \begin{pmatrix} 21 & 28 \\ 56 & 84 \\ 126 & 210 \end{pmatrix} \\ \left. \begin{pmatrix} 1 & 6 \\ 1 & 7 \end{pmatrix}, \begin{pmatrix} 21 & 56 & 126 \\ 28 & 84 & 210 \end{pmatrix}, \begin{pmatrix} 252 & 462 \\ 462 & 924 \end{pmatrix} \right\}.$$

Algorithme :

EF/mat_cell/trans_mat_mat_cell.m

On remet ici la partie déclarative de la fonction :

```
function Sc = trans_mat_mat_cell(S, D, Dnc) ;
```

- Calcul de Nbr qui repère dans S les emplacements des blocs :

```
Nbr = Dnc ;
```

```
%Somme cumulee
```

```
Nbr = cumsum(Nbr) ;
```

```
Nbr = [0,Nbr] ;
```

- Traitement du cas où S est une matrice :

```
%Cas matrice :
```

```
Si (size(S,1)>1) & (size(S,2)>1)
```

```
    %Sc est carree par blocs, de taille, par blocs,
```

```
    % length(Nbr) - 1 :
```

```

    Pour ic=1 :length(Nbr) - 1
        Pour jc=1 :length(Nbr) - 1
            Sc{ic,jc} = ...
                S(1 + Nbr(ic) : Nbr(ic+1), 1 + Nbr(jc) : Nbr(jc+1));
        Fin ;
    Fin ;
• Traitement du cas où S est un vecteur :
Sinon
    %Cas vecteur ligne
    Si size(S,1)==1
        Pour ic=1 :length(Nbr) - 1
            Sc{1,ic} = S(1 + Nbr(ic) : Nbr(ic+1));
        Fin ;
    %Cas vecteur colonne
    Sinon
        Pour ic=1 :length(Nbr) - 1
            Sc{ic,1} = S(1 + Nbr(ic) : Nbr(ic+1));
        Fin ;
    Fin ;
Fin ;

```

Fonction :

```
function S = trans_mat_cell_mat(Sc) ;
```

Transformation d'une matrice de type `CELL ARRAY` en matrice de type `DOUBLE ARRAY` ou `SPARSE ARRAY` par concaténation des cellules.

`Sc` est une matrice de type `CELL ARRAY` dont les cellules sont des matrices de type `DOUBLE ARRAY` ou bien `SPARSE ARRAY`, l'instruction

```
S = trans_mat_cell_mat(Sc) ;
```

renvoie la matrice `S` donnée par :

$$S = \begin{pmatrix} Sc\{1,1\} & \cdots & Sc\{1,j\} & \cdots & Sc\{1,end\} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Sc\{i,1\} & \cdots & Sc\{i,j\} & \cdots & Sc\{i,end\} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ Sc\{end,1\} & \cdots & Sc\{end,j\} & \cdots & Sc\{end,end\} \end{pmatrix}.$$

Les cellules de `Sc` doivent être compatibles en taille pour la concaténation. Si une cellule `Sc{i,j}` est vide elle est supprimée.

Si une des cellules `Sc{i,j}` est de type `SPARSE ARRAY` alors `S` est de type `SPARSE ARRAY`. Sinon `S` est de type `DOUBLE ARRAY`.

Variables d'entrée :

`Sc` : `CELL ARRAY`. Représentation d'une matrice par blocs. Le bloc (i, j) est donné par la matrice `Sc{i,j}` qui est de type `DOUBLE ARRAY` ou `SPARSE ARRAY`.

Variables de sortie :

`S` : `DOUBLE ARRAY`. Si une des cellules `Sc{i,j}` est de type `SPARSE ARRAY` alors `S` est de type `SPARSE ARRAY`. Sinon `S` est de type `DOUBLE ARRAY`.

Exemple : Cet exemple est donné dans le `help` de la fonction `trans_mat_cell_mat`.

EXEMPLE

```
Sc = {[1,2], [1,3];
      [1,3], [5,6]};
S = trans_mat_cell_mat(Sc);
disp(S);
```

qui renvoie la matrice

$$S = \begin{pmatrix} 1 & 2 & 1 & 3 \\ 1 & 3 & 5 & 6 \end{pmatrix}.$$

Algorithme :

EF/mat_cell/trans_mat_cell_mat.m

On remet ici la partie déclarative de la fonction :

```
function S = trans_mat_cell_mat(Sc) ;
```

Initialisation de S :

```
S = [] ;
```

- Cas où Sc a plusieurs lignes :

```
Si size(Sc,1)>1
```

```
    Pour i=1 :size(Sc,1)
```

```
        A = [] ;
```

```
        Pour j=1 :size(Sc,2)
```

```
            %On ne met sur la ligne que des matrices non vides
```

```
            Si ~isempty(Sc{i,j})
```

```
                A = [A,Sc{i,j}] ;
```

```
            Fin ;
```

```
            %Economie de memoire
```

```
            Sc{i,j} = [] ;
```

```
        Fin ;
```

```
        %On peut mettre A = [] : la nouvelle ligne n'est pas creee
```

```
        S = [S;A] ;
```

```
    Fin ;
```

- Cas ou Sc a une seule ligne :

```
Sinon
```

```
    A = [] ;
```

```
    Pour j=1 :size(Sc,2)
```

```
        A = [A,Sc{1,j}] ;
```

```
    Fin ;
```

```
    S = [A] ;
```

```
Fin ;
```

Fonction :

```
function Fc2 = supp_lin_col_vec_cell(Fc, D) ;
```

Suppression de lignes d'un vecteur colonne, dont les blocs sont représentés par les cellules d'un tableau de cellules.

Pour chaque valeur de i entre 1 et le nombre de lignes de `D.inc_point`, alors pour $r = \text{D.inc\_point}\{i, 1\}$ et $m = \text{D.inc\_point}\{i, 2\}$, la commande

```
    Fc2 = supp_lin_col_vec_cell(Sc, D)
```

supprime, si m est un scalaire, la ligne m de $\text{Fc}\{r\}$. Si m est un vecteur ligne ou colonne alors l'élimination se fait pour chaque valeur de m et si $m = \text{'end'}$, alors l'élimination se fait sur la dernière ligne de $\text{Fc}(r)$. Le résultat est mis dans la variable `Fc2`.

Enfin le champ `D.inc_point` peut ne pas être défini ou être vide, dans ce cas `Fc2 = Fc` de manière inchangée.

Variables d'entrée :

`Fc` : CELL ARRAY. Vecteur colonne. Représentation par bloc d'un vecteur colonne dont le bloc i est donné par le vecteur colonne $\text{Fc}\{i\}$, $\text{Fc}\{i\}$ étant de type DOUBLE ARRAY ou SPARSE ARRAY.

`D` : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1. Cette fonction utilise le champ suivant de `D` :

`D.inc_point` : CELL ARRAY. Tableau de 3 colonnes. Les termes de la première colonne sont des scalaires indiquant la cellule sur laquelle on fait l'élimination. Les termes de la seconde colonne sont des vecteurs lignes ou colonnes, ou le terme `'end'`, et dont chaque terme désigne la ligne et la colonne à éliminer. Les termes de la troisième colonne sont des vecteurs lignes ou colonnes dont les valeurs ne sont pas utilisées dans cette fonction. Ils peuvent être mis à `[]`.

On peut aussi poser `D = []` dans cette fonction, le terme `D.inc_point` est facultatif ou peut prendre la valeur `[]`.

Variables de sortie :

`Fc2` : CELL ARRAY. Représentation sous forme de tableau de cellules du vecteur par blocs résultat des éliminations. Les cellules $\text{Fc2}\{i\}$ sont des vecteurs colonnes de même type que $\text{Fc}\{i\}$: DOUBLE ARRAY ou SPARSE ARRAY.

Exemple :

Cet exemple est donné dans le `help` de la fonction `supp_lin_col_vec_cell`.

EXEMPLE

```
clear
```


4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/supp_lin_col_vec_cell`

```

Fc = {[1 :3]';[11 :14]';[21 :24]'};
D.inc_point = {1,1,[]; 2, 'end',[]; 3, [1,2],[]};
Fc2 = supp_lin_col_vec_cell(Fc,D);
disp('Vecteur en entree taille Fc et [Fc{1};Fc{2};Fc{3}] :''')
disp(Fc);
disp([Fc{1};Fc{2};Fc{3}]);
disp('Composantes eliminees');
disp(D.inc_point);
disp('Vecteur en sortie [Fc2{1};Fc2{2};Fc2{3}] :''');
disp([Fc2{1};Fc2{2};Fc2{3}]);

```

Cet exemple transforme la variable

$$Fc = \left\{ \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 11 \\ 12 \\ 13 \\ 14 \end{pmatrix}, \begin{pmatrix} 21 \\ 22 \\ 23 \\ 24 \end{pmatrix} \right\}$$

avec

$$D.inc\_point = \{1, 1, []; 2, 'end', []; 3, [1, 2], []\};$$

en

$$Fc2 = \left\{ \begin{pmatrix} 2 \\ 3 \end{pmatrix}, \begin{pmatrix} 11 \\ 12 \\ 13 \end{pmatrix}, \begin{pmatrix} 23 \\ 24 \end{pmatrix} \right\}.$$

Algorithme :

EF/mat_cell/supp_lin_col_vec_cell.m

On remet ici la partie déclarative de la fonction :

```
function Fc2 = supp_lin_col_vec_cell(Fc, D) ;
```

Suppression dans `Fc` des termes provenant de `D.inc_point`.

- Calcul des variables `D2.inc_point` et `D3.inc_point` qui prennent les mêmes valeurs que dans la fonction `trans_sol_mat_sol_cell`. L'algorithme est identique à l'exception des lignes signalées par % qui indiquent les différences. La troisième colonne des tableaux de cellules ci-dessous, bien que n'intervenant pas dans les calculs, a été conservée.

```
Pour i=1 :size(D.inc_point,1)
    C1(i) = D.inc_point{i,1};
Fin;
[I,J] = sort(C1);
Pour i=1 :size(D.inc_point,1)
    D2.inc_point{i,1} = D.inc_point{J(i),1};
    D2.inc_point{i,2} = D.inc_point{J(i),2};
    D2.inc_point{i,3} = D.inc_point{J(i),3};
Fin;
Pour i=1 :length(Fc)
%(Pour i=1 :length(Dnc) dans trans_sol_mat_sol_cell)
    D3.inc_point{i,1} = i;
    D3.inc_point{i,2} = [];
    D3.inc_point{i,3} = [];
Fin;
Pour i=1 :size(D2.inc_point,1)
    M1 = D2.inc_point{i,1};
    M2 = D2.inc_point{i,2};
    M3 = D2.inc_point{i,3};
    Si ischar(M2)
        Si M2=='end'
            M2=length(Fc{M1,1});
            %(M2 = Dnc(M1) ; dans trans_sol_mat_sol_cell)
        Sinon
            error('Chaine de caractere doit etre egale a end')
        Fin;
    Fin;
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/supp_lin_col_vec_cell`

```
D3.inc_point{M1,2} = [D3.inc_point{M1,2}( : ) ; M2( :)] ;
D3.inc_point{M1,3} = [D3.inc_point{M1,3}( : ) ; M3( :)] ;
Fin ;
```

- Elimination des lignes de `Fc` : par construction de `D3.inc_point` on ne passe qu'une seule fois sur une cellule (nécessaire pour éliminer avec la manière ci-dessous).

```
Pour j=1 :size(D3.inc_point,1)
    r = D3.inc_point{j,1};%Numero de cellule sur laquelle
                        %porte l'elimination
    m = D3.inc_point{j,2};%Colonne de numeros de la ligne dans
                        %la cellule a eliminer
    %Elimination de la ligne
    Fc{r}(m) = [] ;
Fin ;
Fc2 = Fc ;
```

Fonction :

```
function Sc2 = supp_lin_col_mat_cell(Sc, D) ;
```

Suppression de lignes et de colonnes dans des tableaux de type `CELL ARRAY` notamment dans une matrice par blocs rangée dans un tableau de cellules.

Soit `Sc` un tableau de type `CELL ARRAY`. Pour chaque valeur de i allant de 1 au nombre de lignes de `D.inc_point`, soient $r = \text{D.inc\_point}\{i,1\}$ et $m = \text{D.inc\_point}\{i,2\}$, alors si m est un scalaire, la commande

```
Sc2 = supp_lin_col_mat_cell(Sc, D) ;
```

supprime la ligne m de $\text{Sc}\{r,1\}, \dots, \text{Sc}\{r,i\}, \dots, \text{Sc}\{r,\text{end}\}$ et la colonne m de $\text{Sc}\{1,r\}, \dots, \text{Sc}\{i,r\}, \dots, \text{Sc}\{\text{end},r\}$. Si m est un vecteur ligne ou colonne alors l'élimination se fait pour chaque valeur de m . Si $m = \text{'end'}$, alors l'élimination se fait pour la valeur de $\min(\text{size}(\text{Sc}\{r,r\}))$.

Dans le cas où $m = \text{'end'}$ et où `Sc` représente une matrice par blocs (au sens matriciel de la compatibilité lignes et colonnes des blocs) et si $\text{Sc}\{r,r\}$ est une matrice carrée, ce qui est le cas des matrices assemblées dans `EF_assemblage` alors la dernière ligne de $\text{Sc}\{r,1\}, \dots, \text{Sc}\{r,i\}, \dots, \text{Sc}\{r,\text{end}\}$ et la dernière colonne de $\text{Sc}\{1,r\}, \dots, \text{Sc}\{i,r\}, \dots, \text{Sc}\{\text{end},r\}$ sont éliminées.

Dans le cas où $m = \text{'end'}$ on doit avoir $r \leq \min(\text{size}(\text{Sc}))$, le cas $r > \min(\text{size}(\text{Sc}))$ n'est pas traité.

Enfin si le champ `D.inc_point` est soit non défini, soit vide, alors `Sc2 = Sc` en sortie, de manière inchangée.

Variables d'entrée :

`Sc` : `CELL ARRAY`. Tableau dont les blocs (i,j) i.e. $\text{Sc}\{i,j\}$ sont de type `DOUBLE ARRAY` ou `SPARSE ARRAY`. `Sc` est une représentation d'une matrice par blocs si les cellules $\text{Sc}\{i,j\}$ ont la compatibilité suivante :

- à i fixé le nombre de lignes de $\text{Sc}\{i,j\}$ est une constante d_i indépendante de j .
- à j fixé le nombre de colonnes de $\text{Sc}\{i,j\}$ est une constante d_j indépendante de i .

`D` : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1. Cette fonction utilise le champ suivant de `D` :

`D.inc_point` : `CELL ARRAY`. Tableau de 3 colonnes. Les termes de la première colonne sont des scalaires indiquant la cellule sur laquelle on fait l'élimination. Les termes de la seconde colonne sont des vecteurs lignes ou colonnes, ou le terme `'end'`, et dont chaque terme désigne la ligne et la colonne à éliminer. Les termes de la troisième colonne sont des vecteurs lignes ou colonnes dont les valeurs ne sont pas utilisées dans cette fonction. Ils peuvent être mis à `[]`.

On peut aussi poser `D = []` dans cette fonction, le terme `D.inc_point` est facultatif ou peut prendre la valeur `[]`, pour ces deux cas on obtient `Sc2 = Sc`.

Variables de sortie :

Sc2 : CELL ARRAY. Représentation sous forme de tableau de cellules de la matrice par blocs résultat des éliminations. La matrice $\text{Sc2}\{i,j\}$ est du même type que $\text{Sc}\{i,j\}$: soit DOUBLE ARRAY, soit SPARSE ARRAY.

Exemple : Cet exemple est donné dans le `help` de la fonction `supp_lin_col_mat_cell`.

EXEMPLE

```
clear ;
    clear ;
    Sc = ...
{[1 :3 ;3 :5 ;5 :7], [5 :7 ;7 :9 ;9 :11], [9 :11 ;11 :13 ;13 :15] ;
 [2 :4 ;4 :6 ;6 :8], [6 :8 ;8 :10 ;10 :12], [10 :12 ;12 :14 ;14 :16] ;
 [1 :3 ;3 :5 ;5 :7], [5 :7 ;7 :9 ;9 :11], [9 :11 ;11 :13 ;13 :15]} ;
    D.inc_point={1, 1, [] ; 2, 'end', [] ; 3, [1,2], []} ;
    Sc2 = supp_lin_col_mat_cell(Sc, D) ;
    disp('En entree matrice Sc par blocs Sc{i,j}') ;
    disp(Sc) ;
    disp([Sc{1,1},Sc{1,2},Sc{1,3} ;
          Sc{2,1},Sc{2,2},Sc{2,3} ;
          Sc{3,1},Sc{3,2},Sc{3,3}]) ;
    disp('Composantes eliminees') ;
    disp(D.inc_point) ;
    disp('Sortie :matrice par blocs Sc2{i,j} apres eliminations') ;
    disp(Sc2) ;
    disp([Sc2{1,1},Sc2{1,2},Sc2{1,3} ;
          Sc2{2,1},Sc2{2,2},Sc2{2,3} ;
          Sc2{3,1},Sc2{3,2},Sc2{3,3}]) ;
```

Cet exemple transforme

$$\text{Sc} = \left\{ \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{pmatrix}, \begin{pmatrix} 5 & 6 & 7 \\ 7 & 8 & 9 \\ 9 & 10 & 11 \end{pmatrix}, \begin{pmatrix} 9 & 10 & 11 \\ 11 & 12 & 13 \\ 13 & 14 & 15 \end{pmatrix} \right.$$

$$\left. \begin{pmatrix} 2 & 3 & 4 \\ 4 & 5 & 6 \\ 6 & 7 & 8 \end{pmatrix}, \begin{pmatrix} 6 & 7 & 8 \\ 8 & 9 & 10 \\ 10 & 11 & 12 \end{pmatrix}, \begin{pmatrix} 10 & 11 & 12 \\ 12 & 13 & 14 \\ 14 & 15 & 16 \end{pmatrix} \right.$$

$$\left. \begin{pmatrix} 1 & 2 & 3 \\ 3 & 4 & 5 \\ 5 & 6 & 7 \end{pmatrix}, \begin{pmatrix} 5 & 6 & 7 \\ 7 & 8 & 9 \\ 9 & 10 & 11 \end{pmatrix}, \begin{pmatrix} 9 & 10 & 11 \\ 11 & 12 & 13 \\ 13 & 14 & 15 \end{pmatrix} \right\}$$

avec

$$D.inc\_point = \{1, 1, []; 2, 'end', []; 3, [1, 2], []\};$$

en

$$Sc = \begin{Bmatrix} \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix}, & \begin{pmatrix} 7 & 8 \\ 9 & 10 \end{pmatrix}, & \begin{pmatrix} 13 \\ 15 \end{pmatrix} \\ \begin{pmatrix} 3 & 4 \\ 5 & 6 \end{pmatrix}, & \begin{pmatrix} 6 & 7 \\ 8 & 9 \end{pmatrix}, & \begin{pmatrix} 12 \\ 14 \end{pmatrix} \\ (6 & 7), & (9 & 10), & (15) \end{Bmatrix}.$$

Algorithme :

EF/mat_cell/supp_lin_col_mat_cell.m

On remet ici la partie déclarative de la fonction :

```
function Sc2 = supp_lin_col_mat_cell(Sc, D) ;
```

Suppression dans Sc des termes provenant de D.inc_point.

- Calcul des variables D2.inc_point et D3.inc_point qui prennent les mêmes valeurs que dans la fonction `trans_sol_mat_sol_cell`. L'algorithme est identique à l'exception des lignes marquées par % qui indiquent les différences. La troisième colonne es tableaux de cellules ci-dessous, bien que n'intervenant pas dans les calculs, a été conservée.

```
Pour i=1 :size(D.inc_point,1)
```

```
    C1(i) = D.inc_point{i,1} ;
```

```
Fin ;
```

```
[I,J] = sort(C1) ;
```

```
Pour i=1 :size(D.inc_point,1)
```

```
    D2.inc_point{i,1} = D.inc_point{J(i),1} ;
```

```
    D2.inc_point{i,2} = D.inc_point{J(i),2} ;
```

```
    D2.inc_point{i,3} = D.inc_point{J(i),3} ;
```

```
Fin ;
```

```
Pour i=1 :max(size(Sc))
```

```
%(Pour i=1 :length(Dnc) dans trans_sol_mat_sol_cell)
```

```
    D3.inc_point{i,1} = i ;
```

```
    D3.inc_point{i,2} = [] ;
```

```
    D3.inc_point{i,3} = [] ;
```

```
Fin ;
```

```
Pour i=1 :size(D2.inc_point,1)
```

```
    M1 = D2.inc_point{i,1} ;
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `mat_cell/supp_lin_col_mat_cell`

```
M2 = D2.inc_point{i,2} ;
```

```
M3 = D2.inc_point{i,3} ;
```

```
Si ischar(M2)
```

```
    Si M2=='end'
```

- Test sur les dimensions : On doit avoir $\text{all}([M1, M1] \leq \text{size}(\text{Sc}))$, sinon il y a un message d'erreur. On a un **warning** si la matrice $\text{Sc}\{M1, M1\}$ n'est pas carrée.

Suite de l'algorithme :

```
M2 = min(size(Sc{M1,M1}))
```

```
%(M2 = Dnc(M1) ; dans trans_sol_mat_sol_cell)
```

```
Sinon
```

```
    error('Chaine de caractere doit etre egale a end')
```

```
Fin ;
```

```
Fin ;
```

```
D3.inc_point{M1,2} = [D3.inc_point{M1,2}( :) ; M2( :)] ;
```

```
D3.inc_point{M1,3} = [D3.inc_point{M1,3}( :) ; M3( :)] ;
```

```
Fin ;
```

- Fin du calcul de `D2.inc_point` et `D3.inc_point`.
- Elimination des lignes et des colonnes de `Sc` : par construction de `D3.inc_point` on ne passe qu'une seule fois sur une cellule (nécessaire pour éliminer avec la manière ci-dessous).

```
Pour j=1 :size(D3.inc_point,1)
```

```
    r = D3.inc_point{j,1} ; %Numero de cellule sur laquelle
```

```
        %porte l'elimination
```

```
    m = D3.inc_point{j,2} ; %Colonne de numeros de la ligne dans
```

```
        %la cellule a eliminer
```

```
    %Elimination des lignes et colonnes
```

```
    s = size(Sc) ;
```

```
    Pour i=1 :s(2)
```

```
        Si r<=s(1)
```

```
            Sc{r,i}(m, :) = [] ;
```

```
        Fin
```

```
    Fin ;
```

```
    Pour i=1 :s(1)
```

```
        %On a par construction r<=s(1),
```

```
        %Si r > s(2) on ne fait rien sinon cela cree des colonnes vides
```

```
    Si r<=s(2)
        Sc{i,r}( :,m) = [] ;
    Fin ;
Fin ;
Sc2 = Sc ;
```

Fonctions de EF/Assemblage.**Fonction :**

```
function R2 = base_nt(R, X, Y, arb) ;
```

Sur un morceau de la frontière du domaine, calcule, aux *ddl* (degrés de liberté) frontières d'un espace d'EF donné, des vecteurs orthogonaux à la frontière du domaine. En général l'espace d'EF sera un des espaces composant l'espace EF produit E . La direction des vecteurs sera celle de la normale extérieure domaine.

Les variables X, Y sont les coordonnées des sommets du maillage, La variable arb est une variable décrivant des arêtes du maillage et les *ddl* d'un espace d'EF placés sur ces arêtes. La matrice $arb(:, [1, 2])$ décrit les sommets d'arêtes successives qui forment un morceau de frontière γ supposé connexe. Les numéros des *ddl* appartenant à γ sont donnés par le vecteur $arb(:, 4 : \text{end})$. La variable R est un vecteur colonne de longueur au moins égale à $\max(\max(arb(:, 4 : \text{end})))$.

Alors l'instruction

```
R2 = base_nt(R, X, Y, arb) ;
```

effectue le calcul suivant. Soit $1 \leq i \leq \text{length}(R)$, si i n'appartient pas à $arb(:, 4 : \text{end})$, alors i n'est pas un numéro de *ddl* appartenant à γ et on pose :

$$R2(i) = R(i).$$

Maintenant, si i appartient à $arb(:, 4 : \text{end})$ alors i est un numéro de *ddl* noté x tel que $x \in \gamma$ alors :

- si x n'est pas un sommet de triangle ou si x est une des deux extrémités de γ dans le cas où γ n'est pas une courbe fermée, on pose :

$$R2(i) = R(i) + |a|c(n_a),$$

où a est l'arête de γ telle que $x \in a$, $|a|$ sa longueur, n_a la normale unitaire extérieure de γ sur a et $c(n_a)$ la représentation complexe de ce vecteur.

- sinon x est un sommet de triangle et ce sommet est donc commun à deux arêtes (appartenant ou non à un même triangle) frontières a et b de normales extérieures n_a, n_b et de longueurs respectives $|a|, |b|$, on pose :

$$R2(i) = R(i) + |a|c(n_a) + |b|c(n_b).$$

Variables d'entrée :

R : DOUBLE ARRAY (COMPLEX). Vecteur colonne de longueur au moins égale à $\max(\max(\text{arb}(:, 4 : \text{end})))$. En général dans le code cette longueur est égale à la dimension de l'espace d'EF courant auquel appartiennent les *ddl*.

X, Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

arb : DOUBLE ARRAY. Cette variable donne une description d'un morceau de frontière γ connexe du domaine réunion d'arêtes du maillage : description des arêtes et des *ddl* de l'espace d'EF courant, en général un des espaces composant l'espace EF produit E , portés par ces arêtes.

En général dans le code, ces arêtes décrivent un des morceaux qui partitionnent la frontière (cf. le Paragraphe 2.2.2 et la variable **Front** du Tableau 2.2.2-1 de ce Paragraphe). Les arêtes de ce morceau de frontière sont alors numérotées localement sur cette frontière. Dans ce cas **arb**(*i*, :) fait référence à la *i*-ème arête de ce morceau.

- Le terme **arb**(*i*, 1 : 3) est le vecteur ligne des trois numéros de sommets du triangle K qui porte la *i*-ème arête de **arb**. Les deux premiers sommets sont les sommets de l'arête et le troisième celui du sommet intérieur au domaine. L'ordre des sommets est le suivant. Soient a_1, a_2, a_3 les numéros de sommet du triangle K donnés dans leur ordre de numérotation locale sur ce triangle (cette numérotation locale est donnée par le tableau **T** de la structure du maillage du Tableau 2.2.2-1, Paragraphe 2.2.2).

Si, localement sur le triangle, cette arête est l'arête 1 du triangle, alors **arb**(*i*, 1 : 3) = [a_1, a_2, a_3],

si, localement sur le triangle, cette arête est l'arête 2 du triangle, alors **arb**(*i*, 1 : 3) = [a_1, a_3, a_2],

si, localement sur le triangle, cette arête est l'arête 3 du triangle, alors **arb**(*i*, 1 : 3) = [a_2, a_3, a_1].

Le troisième terme **arb**(*i*, 3) donne donc le numéro du sommet du triangle K qui est intérieur au domaine et permet d'orienter la normale vers l'extérieur du domaine.

Ensuite **arb**(*i*, 4 : end) est le vecteur ligne des *ddl* de l'espace d'EF courant.

Le terme **arb**(*i*, 4 : end) est le vecteur nul dans le cas des espaces d'EF discontinus numéros 100 à 199 et 300. Pour les espaces d'EF continus de 1 à 99 et 200, **arb**(*i*, 4 : 5) contient les deux sommets de l'arête *i* dans le même ordre que dans **Am**, **Am** étant la variable décrivant les arêtes calculées par `maillage_triangle_arete_ps`, et **arb**(*i*, 6 : end) contient les autres *ddl* éventuels placés en allant du sommet numéro **arb**(*i*, 4) vers le sommet numéro **arb**(*i*, 5).

Remarque 1. Dans le cas des EF continus **arb**(*i*, 1 : 2) et **arb**(*i*, 4 : 5) sont identiques par construction.

Remarque 2. L'orientation des *ddl* sur `arb` ne joue pas dans le code car cette variable n'est utilisée que pour des conditions de type Dirichlet, où seules les valeurs des fonctions aux *ddl* interviennent.

Variables de sortie :

`R2` : DOUBLE ARRAY (COMPLEX). Vecteur de même taille que `R`. Le terme `R2(i)` est le nombre complexe suivant.

Soit $1 \leq i \leq \text{length}(\mathbf{R})$, si i n'appartient pas à `arb(:, 4 : end)`, alors i n'est pas un numéro de *ddl* appartenant à γ et on pose :

$$\mathbf{R2}(i) = \mathbf{R}(i).$$

Maintenant, si i est un numéro de *ddl* noté x tel que $x \in \gamma$ (alors i appartient à `arb(:, 4 : end)`), alors :

- si x n'est pas un sommet de triangle ou si x est un sommet de triangle mais n'appartient qu'à une seule arête décrite par `arb` (cas où x est une des deux extrémités de γ et que γ n'est pas une courbe fermée), alors :

$$\mathbf{R2}(i) = \mathbf{R}(i) + |a|c(n_a),$$

où a l'arête de γ telle que $x \in a$, $|a|$ sa longueur, n_a la normale unitaire extérieure de γ sur a et $c(n_a)$ la représentation complexe de ce vecteur.

- sinon x est un sommet de triangle et x est commun à deux arêtes a et b , décrites par `arb` et appartenant ou non à un même triangle, de normales extérieures n_a, n_b et de longueurs respectives $|a|, |b|$, on a :

$$\mathbf{R2}(i) = \mathbf{R}(i) + |a|c(n_a) + |b|c(n_b).$$

Le fait que x est une extrémités de γ et que γ n'est pas une courbe fermée se traduit par le fait x appartient à une seule arête décrite par `arb` et que i figure une seule fois dans `arb(:, [1, 2])`. Le fait que x est commun à deux arêtes décrites par `arb` se traduit par le fait que i figure deux fois dans `arb(:, [1, 2])`.

Exemple : Cet exemple est donné par le fichier *script exemple_base_nt.m* du répertoire `EF/exemples/Assemblage`, il donne aussi un exemple d'utilisation de la fonction `front_ddl` qui sera décrite ci-après :

EF/exemples/Assemblage/exemple_base_nt.m

```
%Fichier EF/exemples/Assemblage/exemple_base_nt
%   Utilisation de base_nt : calcul des vecteurs normaux aux DDL
%   du P3 continu sur la frontiere d'un maillage.
%   Commande
```

```

%      cd C :/EF/exemples/Assemblage, exemple_base_nt

clear
close all
%Calcul du maillage : T,X,Y
T = [1,5,4; 1,2,5; 2,6,5; 2,3,6; 4,8,7; 4,5,8; 5,9,8; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;

%Dont la frontiere : Front
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Calcul des DDL de l'espace EF numero 3 (P3 continu)
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, Y, Front) ;
[Td, Ad, Xd, Yd] = maillage_ddl(3, T, Am, Pa, X, Y) ;

%Trace du maillage et des DDL
axis([-0.1,1.1,-0.1,1.1]) ;
for i=1 :size(T,1)
    patch(X(T(i,1 :3)), Y(T(i,1 :3)), 'w', 'Linewidth',1) ;
    hold on ;
    plot(Xd(Td(i, :)), Yd(Td(i, :)), 'ko') ;
end ;

%Numerotation des DDL sur la figure
for i=1 :length(Xd)
    text(Xd(i) + 0.021, Yd(i) + 0.02,num2str(i), 'FontSize', 14) ;
    hold on ;
end

%Calcul des vecteurs orthogonaux aux DDL de la frontiere du domaine

```

```

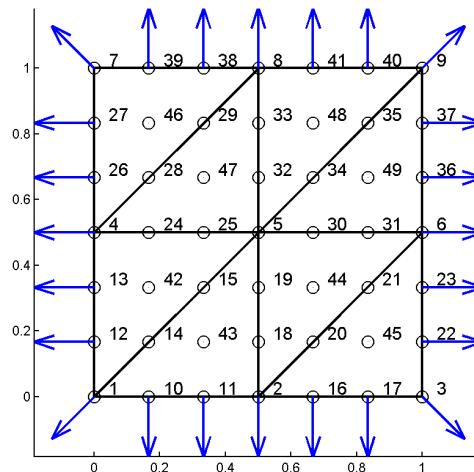
R2 = zeros(length(Xd),1) ;
for j=1 :length(Front)
    arb = front_ddl(T,Ad,Pfc,j) ;
    R2 = base_nt(R2,X,Y,arb) ;
end ;

%Normalisation :
I = find(abs(R2)>0) ;
R2(I) = R2(I)./abs(R2(I)) ;

%Trace des vecteurs
quiver(Xd,Yd,real(R2),imag(R2))
axis([-0.3,1.3,-0.3, 1.3]) ;

%Eventuelle sauvegarde de la figure obtenue : enlever le %
%print('-dpng','C :/EF_text/figures/normales.png') ;

```

Figure 4.3-5. Utilisation des fonctions `base_nt` et `front_ddl`.

Dans cet exemple la frontière est partitionnée en quatre morceaux de frontière avec

```

Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;

```

```
Front{3,1} = [3,2,1] ;
```

```
Front{4,1} = [1,4,7] ;
```

Ensuite, une boucle traite chaque morceau de frontière avec un calcul, avec la fonction `front_ddl`, de la variable `arb` correspondant à ce morceau de frontière, et avec un appel de la fonction `base_nt` :

```
%Calcul des vecteurs orthogonaux aux DDL de la frontiere du domaine
```

```
R2 = [] ;
```

```
R2 = zeros(length(Xd),1) ;
```

```
for j=1 :length(Front)
```

```
    arb = front_ddl(T,Ad,Pfc,j) ;
```

```
    R2 = base_nt(R2,X,Y,arb) ;
```

```
end ;
```

Par exemple, dans cette boucle, pour $j = 1$ les vecteurs aux *ddl* de numéros donnés par [7, 39, 38, 8, 41, 40, 9] valent alors

$$R2([7, 39, 38, 8, 41, 40, 9]) = (1/2i, 1/2i, 1/2i, i, 1/2i, 1/2i, 1/2i).$$

Dans la suite de la boucle, aux coins intersection des morceaux de frontières, la boucle sur `Front` effectue une pondération des normales par la taille des arrêtes. Il est à noter qu'une boucle de ce type est effectuée dans la fonction `changt_base_nt` de changement de base qui pondère les normales.

On normalise ensuite `R2` avec :

```
%Normalisation :
```

```
I = find(abs(R2)>0) ;
```

```
R2(I) = R2(I)./abs(R2(I)) ;
```

Une pondération aux coins identique sera réalisée en prenant pour `Front` :

```
Front{1,1} = [7,8,9,6,3,2,1,4,7] ;
```

et en remplaçant la boucle ci-dessus de calcul de `R2` par :

```
R2 = zeros(length(Xd),1) ;
```

```
arb = front_ddl(T,Ad,Pfc,1) ;
```

```
R2 = base_nt(R2,X,Y,arb) ;
```

ce qui conduit au même résultat pour la valeur de `R2`. Cet exemple est donné par le fichier *script exemple2_base_nt.m* du répertoire `EF/exemples/Assemblage`.

Algorithme :

EF/Assemblage/base_nt.m

On remet ici la partie déclarative de la fonction :

```
function R2 = base_nt(R, X, Y, arb) ;
```

Ci-dessous `pt(1)`, `pt(2)` et `pt(2)` sont les numéros des sommets du triangle qui porte l'arête frontière courante avec `pt(1)`, `pt(2)` numéros de sommets de l'arête et `pt(3)` numéro du point intérieur au domaine. Ensuite `v1` est un vecteur colinéaire à l'arête courante et `v3` un vecteur qui pointe vers l'intérieur du domaine.

```
Pour i=1 :size(arb,1)
```

```
    pt = arb(i, :) ;
```

```
    %Calcul d'un vecteur tangent a l'arete
```

```
    v1 = X(pt(2)) - X(pt(1)) + sqrt(-1)*( Y(pt(2)) - Y(pt(1)) ) ;
```

```
    %Calcul d'un vecteur qui pointe vers l'interieur du domaine :
```

```
    v2 = X(pt(3)) - X(pt(1)) + sqrt(-1)*( Y(pt(3)) - Y(pt(1)) ) ;
```

```
    %Argument dans ]-pi,pi] de v1
```

```
    theta = angle(v1) ;
```

```
    %La normale exterieure et v1 sont dans le sens direct
```

```
    Si angle(v2/v1) >= 0
```

```
        %on ne fait rien ici
```

```
    %v1 et la normale exterieure sont dans le sens direct
```

```
    Sinon
```

```
        theta = angle(-v1) ;
```

```
        v1 = -v1 ;
```

```
    Fin ;
```

```
    %Calcul des vecteurs orthogonaux sur les DDL :
```

```
    Si length(pt)>3 & all(pt(4 :end))==1%il y a des DDL
```

```
        R(pt(4 :end)) = R(pt(4 :end)) + ...
```

```
            abs(v1)*exp(-sqrt(-1)*pi/2)*(v1/abs(v1)) ;
```

```
    Fin ;
```

```
Fin ;
```

```
R2 = R ;
```

Fonction :

```
function arb = front_ddl(T, Ad, Pfc, j) ;
```

Calcul d'un tableau de description des arêtes d'un morceau de frontière et de leur *ddl*.

La commande

```
arb = front_ddl(T, Ad, Pfc, j) ;
```

calcule en fonction de **T** le tableau des sommets de triangles, **Ad** le tableau des *ddl* sur les arêtes décrits arêtes par arêtes et **Pfc** le tableau de correspondance entre les arêtes de la frontière et les triangles qui les portent, la variable **arb** relative au arêtes du morceau de frontière numéro **j**. Pour l'arête de numéro local *i* sur le morceau de frontière **j**, le terme **arb**(*i*, [1, 2, 3]) est le vecteur des numéros de sommets du triangle qui porte cette arête, en commençant par les sommets situés sur la frontière, puis **arb**(*i*, 4 : end) désigne les *ddl* placés sur cette arête avec la valeur du vecteur nul lorsqu'il n'y a pas de *ddl* comme dans le cas des espaces d'EF discontinus de 100 à 199 et 300 qui ont leurs *ddl* intérieurs aux triangles. L'ordre des numéros de sommets et des *ddl* dans **arb**(*i*, :) est précisé dans la description de **arb** à la fonction **base_nt**.

Variables d'entrée :

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Ad : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes) × (Nombre de *ddl* sur les arêtes, sommets compris) avec une seule colonne (nulle) s'il n'y pas de *ddl* sur les arêtes. Description des *ddl* arête par arête. Le terme **Ad**(*i*, :) vaut 0 s'il n'y a pas de *ddl* d'arêtes comme pour les espaces d'EF discontinus. Sinon, **Ad**(*i*, [1, 2]) prend les valeurs des sommets de l'arête *i* (ces valeurs sont données par **Am**(*i*, [1, 2]), où **Am** est la variable de description des arêtes des sommets du maillage calculée par la fonction **maillage_triangle_arete_ps**). Ensuite, s'il y a des *ddl* sur cette arête privée de ses extrémités, **Ad**(*i*, [3, ...]) prend les valeurs des numéros de ces *ddl* en les décrivant consécutivement du sommet d'arête **Ad**(*i*, 1) vers le sommet d'arête **Ad**(*i*, 2). Cette variable ne sert que dans la fonction **cb** pour les conditions de type Dirichlet où l'ordre de numérotation des *ddl* n'est pas utilisé.

Pfc : CELL ARRAY. Vecteur colonne de cellules de longueur égale au nombre de morceaux de frontière du domaine. Cette variable a donc le même nombre de cellules que la variable de maillage **Front** (cf. Tableau 2.2.2-1, Paragraphe 2.2.2). Sur chaque morceau de frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur ce morceau de frontière. La cellule **Pfc**{*i*} est un tableau de trois colonnes de type DOUBLE ARRAY qui décrit les arêtes du morceau de frontière numéro *i* de la manière suivante. Soit une arête de numéro *j* dans sa numérotation sur le morceau de frontière numéro *i*, alors **Pfc**{*i*}(*j*, 1) est le numéro de triangle auquel appartient cette arête, **Pfc**{*i*}(*j*, 2) son numéro d'arête

localement sur ce triangle (valeur parmi 1, 2 ou 3) et $\text{Pfc}\{i\}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes. Pf vaut `[]` si `Front = []`. Le calcul de cette variable est réalisé par `maillage_triangle_arete_ps` à partir de `Front` dans l'ordre des arêtes de `Front{1}`, `Front{2}` *etc.* Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`arb` : `DOUBLE ARRAY`. Variable décrivant les `ddl` pour lesquels on calcule le vecteur orthogonal. Cette variable est décrite dans la fonction `base_nt` précédente.

Exemple : Un exemple d'utilisation de cette fonction est donné dans l'exemple donné pour la fonction `base_nt`.

Algorithme :

`EF/Assemblage/front_ddl.m`

On remet ici la partie déclarative de la fonction :

```
function arb = front_ddl(T, Ad, Pfc, j) ;
```

Soit l'arête numéro k du morceau de frontière numéro j . Cette arête est donc située sur le triangle numero $n_K = \text{Pfc}\{j\}(k, 1)$ et son numéro local d'arête est $n_A = \text{Pfc}\{j\}(k, 2)$.

Le terme $\mathbf{z}(k, :)$ ci-dessous est formé des trois numéros de sommets du triangle, avec dans l'ordre, les deux sommets de l'arête puis le sommet intérieur au domaine. Dans $\mathbf{z}(k, :)$, l'ordre des sommets de l'arête est défini par sa position locale sur le triangle de la manière suivante :

si $\text{T}(n_K, [1, 2, 3]) = [a_1, a_2, a_3]$ alors

- $\mathbf{z} = [a_1, a_2]$ si $n_A = 1$,

- $\mathbf{z} = [a_1, a_3]$ si $n_A = 2$,

- $\mathbf{z} = [a_2, a_3]$ si $n_A = 3$.

En principe, dans le code l'orientation n'a pas d'importance (utilisation principalement dans la fonction `cb`).

Il est à remarquer que l'ordre des sommets est le même que celui rencontré dans la variable `Am` calculée dans `maillage_triangle_arete_ps`. Ceci est une conséquence de l'unique possibilité de choix de `Am` dans l'algorithme de `maillage_triangle_arete_ps` lorsqu'une arête n'est portée que par un seul triangle, comme ici pour une arête frontière.

Ci-dessous

k est le numéro d'arête courante sur le morceau de frontière j ,

$n_K = \text{Pfc}\{j\}(k, 1)$ le numéro de triangle qui porte l'arête courante,

$n_A = \text{Pfc}\{j\}(k, 2)$ le numéro local de l'arête sur le triangle.

Les sommets de triangles portant l'arête courante sont alors $T(n_K, [1, 2, 3])$, avec 3 ordres possibles $T(n_K, [1, 2, 3])$ si $n_A = 1$, $T(n_K, [1, 3, 2])$ si $n_A = 2$, $T(n_K, [2, 3, 1])$ si $n_A = 3$:

```

z = [] ;
Pour k=1 :size(Pfc{j},1) ;
    Si Pfc{j}(k,2)==1
        z(k, :) = T(Pfc{j}(k,1),[1,2,3]) ;
    fin ;
    Si Pfc{j}(k,2)==2
        z(k, :) = T(Pfc{j}(k,1),[1,3,2]) ;
    Fin ;
    Si Pfc{j}(k,2)==3
        z(k, :) = T(Pfc{j}(k,1),[2,3,1]) ;
    Fin ;
Fin ;

arb = [z,(Ad(Pfc{j})( :,3), :))] ;

```

Fonction :

```
function P2 = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, ...
                             i1, i2, Fr, varargin) ;
```

Calcul de matrice de changement de base relative. La commande

```
P2 = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, i1, i2, Fr) ;
```

calcule une matrice de changement de base composée de rotations élémentaires. Le terme P2 est la matrice identité dont les termes suivants sont modifiés. Soient E_{i_1} , E_{i_2} le i_1 -ème et i_2 -ème espace d'EF de l'espace EF produit d'approximation E . Les espaces (E_{i_1}, E_{i_2}) doivent être des espaces identiques). Soit i le numéro d'un *ddl* de l'espace numéro E_{i_1} situé sur un morceau de frontière dont le numéro est dans **Fr**. Soit (n, t) la normale unitaire extérieure à la frontière en ce *ddl* et t la tangente unitaire telle que (n, t) soit dans le sens trigonométrique direct.

Soient M_1, M_2 les numéros respectifs des fonctions de base de E_{i_1} et E_{i_2} relatives à ce *ddl* (dans la numérotation globale des fonctions de base de l'espace EF produit E). On note F et G ces deux fonctions de base.

Soit la rotation d'angle θ qui fait passer de la base $(1, 0), (0, 1)$ à la base (n, t) alors P2 prendra les valeurs

$$\begin{aligned} \text{P2}(M_1, M_1) &= \cos \theta, \\ \text{P2}(M_2, M_1) &= \sin \theta, \\ \text{P2}(M_1, M_2) &= -\sin \theta, \\ \text{P2}(M_2, M_2) &= \cos \theta. \end{aligned}$$

Sur l'espace produit $E_{i_1} \times E_{i_2}$, cela revient donc de passer des fonctions $F(1, 0), G(0, 1)$ éléments de la base de $E_{i_1} \times E_{i_2}$ aux deux nouvelles fonctions de base :

$$\cos(\theta)F(1, 0) + \sin(\theta)G(0, 1) = (\cos(\theta)F, \sin(\theta)G)$$

et

$$-\sin(\theta)F(1, 0) + \cos(\theta)G(0, 1) = (-\sin(\theta)F, \cos(\theta)G).$$

Ensuite, dans la nouvelle base de l'espace EF produit E , la M_1 -ième fonction de base donnée par la fonction $(0, \dots, F, \dots, 0)$ avec F placé sur le M_1 -ième terme est remplacée par la fonction de base donnée par la fonction

$$(0, \dots, \cos(\theta)F, \dots, \sin(\theta)G, \dots, 0)$$

dont les termes non nuls sont placés sur le M_1 -ième et M_2 -ième terme.

La M_2 -ième fonction de base donnée par la fonction $(0, \dots, G, \dots, 0)$ avec G placé sur le M_2 -ième terme est remplacée par la fonction de base donnée par la fonction

$$(0, \dots, \sin(\theta)F, \dots, -\cos(\theta)G, \dots, 0)$$

dont les termes non nuls sont placés sur le M_1 -ième et M_2 -ième terme.

La commande

```
P2 = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, i1, ...
                                     i2, Fr, An) ;
```

permet, avec la variable optionnelle **An**, de donner une priorité pour le choix d'une normale en un point à l'intersection de deux morceaux de frontière, par exemple lorsque ce point est un coin (point anguleux). Cette variable **An** représente la valeur $\text{sb.un}\{i, 4\}$ dans le Tableau 2.2.4-2c du Paragraphe 2.2.4. La variable **An** est une matrice carrée de la taille de **Fr**. La normale choisie à l'intersection de deux frontières de numéros f_1 et f_2 est une pondération de chacune des normales par $\text{An}(f_1, f_2)$ pour le morceau de frontière numéro f_1 et $\text{An}(f_2, f_1)$ pour le morceau de frontière numéro f_2 . Par exemple, si $\text{An}(1, 2) = 1$ et $\text{An}(2, 1) = 0$, c'est la normale du morceau de frontière numéro 1 qui est choisie à l'intersection des morceaux de frontière numéros 1 et 2.

On pourra poser $\text{An} = []$, si l'on impose pas de priorité frontière.

La commande

```
P2 = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, i1, ...
                                     i2, Fr, An, theta) ;
```

permet de réaliser le changement de base avec le couple de vecteur (n', t') obtenu par rotation supplémentaire de **theta** de (n, t) au lieu du couple (n, t) .

Les commandes

```
[P2,R] = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, ...
                        i1, i2, Fr, An, theta) ;
```

ou

```
[P2,R,I] = changt_base_nt(P, T, Xc, Yc, Arc, Pfc, Dnc, D, ...
                        i1, i2, Fr, An, theta) ;
```

avec **An** et **theta** optionnelles permettent de récupérer en sortie le vecteur **R** complexe et, pour la deuxième, commande, le vecteur colonne **I** qui représente l'ensemble des numéros de *ddl* appartenant à la réunion des morceaux de frontière de numéros donnés par les composantes de **Fr** et pour lesquels on a effectué le changement de base. Ainsi le terme $\text{I}(k)$ désigne un *ddl* de E_{i_1} placés sur un des morceaux de frontières appartenant à **Fr**. Le vecteur **R** est alors le vecteur colonne de longueur $\dim(E_{i_1})$ tel que $\text{R}(\text{I})$ donne les valeurs successives des normales unitaire sur les *ddl* décrits par **I** : sur le *ddl* numéro $\text{I}(k)$, on effectue un changement de base relatif aux fonctions de base de E_{i_1} et E_{i_2} lié a ce *ddl* et la normale unitaire extérieure vaut $\text{R}(\text{I}(k))$ sur ce *ddl*.

Variables d'entrée :

P : `DOUBLE ARRAY`. Matrice carrée de la dimension de l'espace EF produit E (qui vaut `sum(Dnc)`). Matrice initiale de changement de base. Cette matrice peut être égale à l'identité, ou indiquer déjà un changement de base pour d'autres composantes de l'espace EF produit que celles concernées par le changement de base.

T : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Xc, **Yc** : `CELL ARRAY`. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type `DOUBLE ARRAY`. Les vecteurs $Xc\{i\}$ et $Yc\{i\}$ sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Arc : `CELL ARRAY`. Vecteur ligne. $Arc\{i\}$ est une matrice de type `DOUBLE ARRAY` décrivant les *ddl* arêtes par arêtes du i -ème type d'espace d'EF choisi, *cf.* le Tableau 2.2.3-1b du Paragraphe 2.2.3, $Arc\{i\}$ est un vecteur colonne nul dans le cas d'un espace d'EF qui n'a pas de *ddl* d'arêtes comme les espaces d'EF discontinus de numéros 100 à 199 et de numéro 300. La valeur de $Arc\{i\}$ est donnée par la valeur de la variable `Ad` calculée dans `maillage_ddl` dans le cas du i -ème type d'espace d'EF choisi.

Pfc : `CELL ARRAY`. Vecteur colonne de cellules de longueur égale au nombre de morceaux de frontière du domaine. Cette variable a donc le même nombre de cellules que la variable de maillage `Front` (*cf.* Tableau 2.2.2-1, Paragraphe 2.2.2). Sur chaque morceau de frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur ce morceau de frontière. La cellule $Pfc\{i\}$ est un tableau de trois colonnes de type `DOUBLE ARRAY` qui décrit les arêtes du morceau de frontière numéro i de la manière suivante. Soit une arête de numéro j dans sa numérotation sur le morceau de frontière numéro i , alors $Pfc\{i\}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $Pfc\{i\}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3) et $Pfc\{i\}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes. `Pf` vaut `[]` si `Front` = `[]`. Le calcul de cette variable est réalisé par `maillage_triangle_arete_ps` à partir de `Front` dans l'ordre des arêtes de `Front{1}`, `Front{2}` *etc.* Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Dnc : `DOUBLE ARRAY`. Vecteur ligne. $Dnc(i)$ est la dimension du i -ème espace d'EF de l'espace EF produit E .

D : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1. Dans cette fonction seul le champ `D.f` est utilisé avec :

D.f : `DOUBLE ARRAY`. Vecteur ligne. $D.f(i)$ est un numéro du choix d'espace dans la liste donnée par le vecteur `D.E` : le i -ème espace de l'espace EF produit E a pour numéro $D.E(D.f(i))$.

`i1, i2 : DOUBLE ARRAY`. Position dans l'espace produit EF des deux espaces d'EF sur lesquels on effectue un changement de base.

`Fr : DOUBLE ARRAY`. Vecteur ligne des numéros de morceaux de frontière qui interviennent dans le changement de base.

`varargin{1} : DOUBLE ARRAY`. Matrice `An` vue ci-dessus indiquant les priorités de frontière dans les coins. On pourra poser `varargin{1} = []`, si on n'impose pas de priorité frontière.

`varargin{2} : DOUBLE ARRAY`. Valeur de l'angle θ pour la rotation suivante. Lorsque `varargin{2}` est utilisée, c'est le couple de vecteur (n', t') obtenu par rotation θ de (n, t) qui est utilisé pour le changement de base au lieu de (n, t) . On peut poser `varargin{2} = []` pour $\theta = 0$.

Variables de sortie :

`P2 : DOUBLE ARRAY`. Matrice carrée de la dimension de l'espace EF produit E (qui vaut `sum(Dnc)`). Nouvelle matrice de changement de base.

`varargout{1} : DOUBLE ARRAY (COMPLEX)`. Noté ici R . Le terme R est le vecteur colonne de longueur $\dim(E_{i_1})$ tel que $R(d)$ est la valeur de la normale unitaire portée par le *ddl* numéro d de E_{i_1} si celui-ci est sur un des morceaux de frontière dont le numéro est une composante de `Fr`, et vaut 0 sinon.

`varargout{2} : DOUBLE ARRAY`. Noté ici I . Vecteur colonne des numéros de *ddl* appartenant à la réunion des morceaux de frontière de numéros donnés par les composantes de `Fr`. Le terme $R(I(k))$ est alors la valeur de la normale unitaire extérieure portée par le *ddl* numéro $I(k)$.

Exemple : L'exemple suivant est donné par le fichier *script exemple_changt_base_nt.m* du répertoire EF/exemples/Assemblage qui est aussi l'exemple donné dans le `help` de `base_nt`.

Cet exemple porte sur le maillage de la Figure 4.3-5. Tout d'abord on considère un premier cas avec un changement de base sur le morceau frontière numéro 1 ($y = 1$). Ceci est effectué par la commande :

```
%Cas 1) Frontiere 1
```

```
...
```

```
Fr = 1 ;
```

```
[P2,R2,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, D, 1, 2, Fr) ;
```

Sur cette frontière le passage de la base canonique de $\mathbb{R}^2$ à la base (n, t) (normale unitaire extérieure et tangente, tel que (n, t) soit dans le sens direct) s'obtient donc par rotation de $\frac{\pi}{2}$ de la base canonique. Les normales considérées sont données sur le premier graphe de la Figure 4.3-6. La matrice de changement de base calculée est donc notée `P2`.

Ensuite, on considère second cas avec un changement de base sur toute la frontière en utilisant la variable `An` décrite ci-dessus pour établir une priorité (par un choix de pondération de 1 pour un morceau de frontière et de 0 pour l'autre morceau) pour le choix des normales dans les coins : la frontière numéro 1 ($y = 1$) est prioritaire sur les frontières numéros 2 ($x = 1$) et 4 ($x = 0$), ensuite la frontière numéro 2 est prioritaire sur la frontière numéro 3 ($y = 0$) et la frontière numéro 3 est prioritaire sur la frontière numéro 4. Les normales considérées sont celles du second graphe de la Figure 4.3-6. La matrice de changement de base calculée est notée `P3`. Les commandes coorepondnates sont données par :

```
%Cas 2) Autre exemple avec priorite 1 sur 2 et 4, 2 sur 3
%   et 3 sur 4 :
Fr = [1,2,3,4] ;
An = [0 1 0 1 ;
      0 0 1 0 ;
      0,0,0,1 ;
      0,0,0,0] ;

%Matrice de passage relative aux DDL P3 des morceaux de
%   frontiere 1, 2, 3, 4 :
[P3,R3,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
                           D, 1, 2, Fr, An) ;
```

Enfin, on considère un troisième cas, identique au cas précédent, mais avec `An = []`. La pondération des normales aux coins est alors celle effectuée par la fonction `base_nt` qui est appelée par `changt_base_nt`, cette pondération est donc réalisée en fonction de la longueur des arêtes qui forment le coin, les vecteurs obtenus par `base_nt` sont ensuite normalisés dans la fonction `changt_base_nt`. Ce choix correspond à celui du Paragraphe 3.6.5 pour le choix de la normale et de la tangente dans le cas d'un point anguleux. Les commandes correspondantes sont données par :

```
%Cas 3) ponderation aux coins :
[P4,R4,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
                           D, 1, 2, Fr, []) ;
```

Dans cet exemple, l'espace produit est le $P_3^c \times P_3^c \times P_1^c$ et on effectue les rotations sur les deux premiers espaces.

Dans le premier cas, les fonctions de bases concernées ont leurs numéros donnés par $v = [7, 8, 9, 38, 39, 40, 41]$ qui sont donc les numéros de *ddl* de la figure 4.3-5 et les numéros $49 + [7, 8, 9, 38, 39, 40, 41]$ (car 49 est la dimension du P_3^c de cet exemple).

La matrice extraite de la matrice de passage P2 est donnée par :

$$P2([v, v + 49], [v, v + 49]) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Les normales obtenues pour les deux premiers cas sont représentées à la figure 4.3-6, ci-dessous, et celles obtenues pour le troisième cas sont identiques à celles de la figure 4.3-5 ci-dessus. Dans le troisième cas on trouve, par exemple :

$$\begin{pmatrix} P4(1, 1) & P4(1, 50) \\ P4(50, 1) & P4(50, 50) \end{pmatrix} = \begin{pmatrix} \cos \frac{5\pi}{4} & -\sin \frac{5\pi}{4} \\ \sin \frac{5\pi}{4} & \cos \frac{5\pi}{4} \end{pmatrix}$$

qui correspond à l'angle de $\frac{5\pi}{4}$ formé par la normale avec l'axe Ox au *ddl* numéro 1 de coordonnées $(0, 0)$.

Les figures correspondant aux deux premiers cas sont les suivantes.

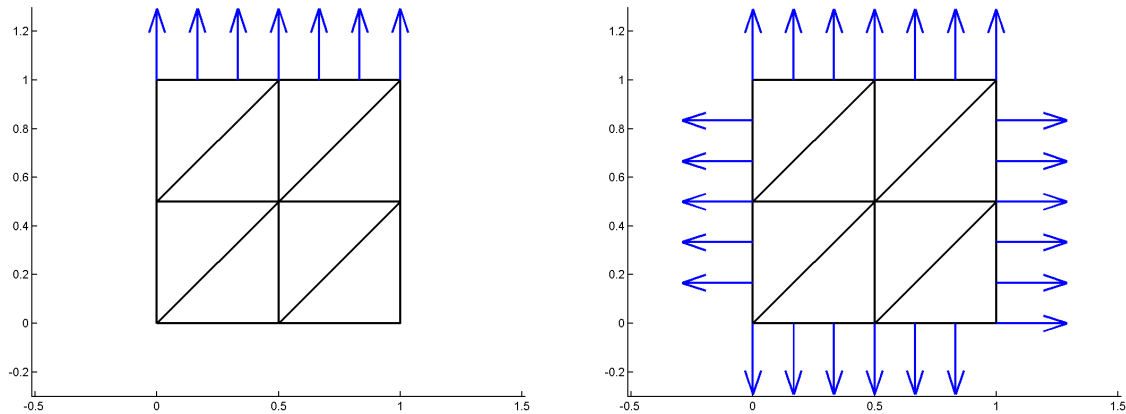


Figure 4.3-6. Utilisation de la fonction `changt_base_nt`.

Le fichier de commandes exemple est le suivant :

EF/exemples/Assemblage/exemple_changt_base_nt.m

```
%Fichier EF/exemples/Assemblage/exemple_changt_base_nt.m
%Utilisation de changt_base_nt l'espace EF choisi etant l'espace
%
%   E =   E1 x E2 x E3 = P3 x P3 x P1 continu
%
%   1) Calcul de la matrice composee de rotations elementaires
%   correspondant au changement de base pour les fonctions de bases
%   de E1 x E2 attaches aux DDL de la frontiere numero 1 du
%   maillage.
%
%   2) Calcul de la matrice de changement de base relativement
%   relatif aux DDL des 4 frontieres partitionnant la frontiere,
%   avec un choix de ponderation de la normale au coins.
%
%   3) Meme calcul que precedemment, sans choix de ponderation des
%   normales par An. La ponderation est alors celle realisee par
%   base_nt lors de son appel par changt_base_nt, puis par
%   normalisation des vecteurs orthogonaux obtenus.
%
%   Dans les trois cas, le changement de base a lieu sur la
%   base de E1 x E2.
%
%Commande :
%
%   cd C :/EF/exemples/Assemblage, exemple_changt_base_nt

clear
close all
%Choix des types d'espaces : P1 et P3
D.E = [1,3] ;
%Choix de l'espace produit P3 x P3 x P1
```

```

D.f = [2,2,1] ;
%Calcul du maillage : T,X,Y
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;

%et la frontiere : Front
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation de Xc, Yc, Tc, Arc, Ed :
donnees.D = D ;
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, X, Y, T, Front) ;

%A titre informatif calcul de arb pour le P1 :
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, Y, Front) ;
arb = front_ddl(T, Arc{2}, Pfc, 1) ;
disp('arb :')
disp(arb) ;

%Dimensions des espaces composant l'espace EF P3 x P3 x P1 :
Dnc = Ed(D.f) ;

%Matrice de passage a modifier :
P = eye(sum(Dnc)) ;

%Cas 1) Frontiere 1
%   P2 : matrice de changement de base relative aux DDL du P3 du
%   morceau de frontiere 1, P3 est composee de rotations
%   elementaire,
%   R2 : coordonnees des vecteur normaux intervenant dans le chan-
%   gement de base et qui definissent les rotation elementaires,

```

4.3. DESCRIPTION DES FONCTIONS

FONCTION Assemblage/changt_base_nt

```
% I : numeros des DDL du P3 pour lesquels on effectue un chan-
% gement de base : I = [7 ; 8 ; 9 ; 38 ; 39 ; 40 ; 41] ;
Fr = 1 ;
[P2,R2,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, D, 1, 2, Fr) ;

%Affichage a EPS pres des matrices de rotation (pi/2), uniquement
% pour les fonctions de bases concernees (round permet de
% condenser l'affichage) :
%v = [7,8,9,38,39,40,41] ;
v = I' ;
disp('P3 extraits :') ;
disp(round(P2([v, v + 49],[v, v + 49])));

%Trace du maillage et des vecteur normaux intervenant dans le
% changement de base
axis([-0.4,1.4,-0.4,1.4]) ;
for i=1 :size(T,1)
    patch(X(T(i,1 :3)), Y(T(i,1 :3)), 'w') ;
    hold on ;
end ;
quiver(Xc{2}(v),Yc{2}(v),real(R2(v)),imag(R2(v)))

%Cas 2) Autre exemple avec priorite 1 sur 2 et 4, 2 sur 3
% et 3 sur 4 :
Fr = [1,2,3,4] ;
An = [0 1 0 1 ;
      0 0 1 0 ;
      0,0,0,1 ;
      0,0,0,0] ;

%Matrice de passage relative aux DDL P3 des morceaux de
% frontiere 1, 2, 3, 4 :
[P3,R3,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
                           D, 1, 2, Fr, An) ;
```

```

%Trace du maillage et des vecteur normaux intervenant dans le
%  changement de base
figure(2)
axis([-0.4,1.4,-0.4,1.4]);
for i=1 :size(T,1)
    patch(X(T(i,1 :3)), Y(T(i,1 :3)), 'w');
    hold on;
end;
quiver(Xc{2}(I),Yc{2}(I),real(R3(I)),imag(R3(I)));

%Cas 3) La ponderation aux coins est celle de base_nt :
[P4,R4,I] = changt_base_nt(P, T, X, Y, Arc, Pfc, Dnc, ...
                           D, 1, 2, Fr, []);

figure(3)
axis([-0.4,1.4,-0.4,1.4]);
for i=1 :size(T,1)
    patch(X(T(i,1 :3)), Y(T(i,1 :3)), 'w');
    hold on;
end;
quiver(Xc{2}(I),Yc{2}(I),real(R4(I)),imag(R4(I)));
%Eventuelle sauvegarde de la figure obtenue : enlever le %
%figure(1)
%print('-dpng','C :/EF_text/figures/exemple_changt_base_nt1.png');
%figure(2)
%print('-dpng','C :/EF_text/figures/exemple_changt_base_nt2.png');
%figure(3)
%print('-dpng','C :/EF_text/figures/exemple_changt_base_nt3.png');

```

Algorithme :

EF/Assemblage/changt_base_nt.m

On remet ici la partie déclarative de la fonction :

```
[P2, varargout] = changt_base_nt(P, T, X, Y, Arc, Pfc, ...
                                Dnc, D, i1, i2, Fr, varargin) ;
```

- Le changement de base a lieu pour deux espaces identiques : on doit avoir $D.f(i1) = D.f(i2)$.

- Calcul de **arb** par **front_ddl** pour chaque numéro de frontière de **Fr**. Ci-dessous, $Arb\{i, j\}$ contient la valeur de la variable **arb** décrivant les sommets du triangle d'appartenance et ses *ddl* pour chaque arête de l'espace d'EF E_{i_1} , i_1 -ème espace de l'espace produit EF courant E , avec $i_1 = i1$:

```
i = D.f(i1) ;
```

```
Pour j = Fr
```

```
    arb = front_ddl(T, Arc{i}, Pfc, j) ;
```

```
    Arb{i,j} = arb ;
```

```
Fin ;
```

- L dimension de E_{i_1} et R vecteur nul de E_{i_1} :

```
L = Dnc(i1) ;
```

```
R = zeros(L,1) ;
```

- Valeurs de l'angle θ de rotation supplémentaire appliqué à (n, t) (*cf.* tableau 2.2.4-2c du Paragraphe 2.2.4) et de la matrice **An** vus ci-dessus :

```
Si nargin < 12
```

```
    theta = 0 ;
```

```
Fin ;
```

```
Si nargin >= 12
```

```
    An = varargin{1} ;
```

```
    theta = 0 ;
```

```
Fin ;
```

```
Si nargin == 13
```

```
    theta = varargin{2} ;
```

```
    si isempty(theta)
```

```
        theta = 0 ;
```

```
    Fin ;
```

```
Fin ;
```

- Calcul des vecteurs normaux sur les *ddl* de E_{i_1} pour chaque numéro de frontière de *Fr*, avec pondération aux sommets communs à deux morceaux de frontière :

```
if nargin < 12 | isempty(An)
```

```
    %1) Calcul de vecteurs orthogonaux a la frontiere :
```

```
    Pour j=1 :length(Fr)
```

```
        R = base_nt(R, X, Y, Arb{D.f(i1), Fr(j)});
```

```
    Fin;
```

```
    I = find(R);
```

```
    %2) Calcul de vecteurs normaux sur les frontiere mis dans R
```

```
    R(I) = R(I)./abs(R(I));
```

```
Fin;
```

- Si *An* existe et est non vide, traitement des sommets communs aux frontières pour le calcul des normales aux coins.

```
Si (nargin>=12)&(~isempty(An))
```

- Calcul par cellules des vecteurs normaux sur les *ddl* de E_{i_1} pour chaque numéro de frontière de *Fr* : le résultat est mis dans la cellule *Rc{i}*.

```
    %Calcul des vecteurs normaux sur les DDL de Ei1 pour chaque
```

```
    %   numero de frontiere de Fr :
```

```
    %1) Initialisations
```

```
    Pour j=1 :max(Fr)
```

```
        Rc{j} = zeros(L,1);
```

```
    Fin;
```

```
    Pour j=1 :length(Fr)
```

```
    %2) Calcul de vecteurs orthogonaux a la frontiere :
```

```
        Rc{Fr(j)} = base_nt(Rc{Fr(j)}, X, Y, Arb{D.f(i1),Fr(j)});
```

```
        R = R + Rc{Fr(j)};
```

```
    Fin;
```

```
    %3) Calcul de vecteurs normaux sur les frontieres
```

```
    Pour j=1 :length(Fr)
```

```
        Ic{Fr(j)} = find(Rc{Fr(j)});
```

```
    Fin;
```

```
    I = find(R);
```

```
    R(I) = R(I)./abs(R(I));
```

```
    %Fin calcul des normales
```

- Calcul de $Acom$, tel que $Acom(Fr(i), Fr(j))$ est la valeur d'un éventuel premier sommet commun au frontière de numéros $Fr(i)$ et $Fr(j)$ avec la valeur nulle si les frontières sont vides. Le terme $Bcom$ est sur le même principe mais relativement à la prise en compte d'un deuxième sommet commun.

```
%Initialisations (necessaire au cas ou zcom soit vide) :
Acom = zeros(max(Fr),max(Fr)) ;
Bcom = zeros(max(Fr),max(Fr)) ;
Pour icom=1 :length(Fr) ;
    Pour jcom =1 :length(Fr) ;
        Si jcom~=icom
            zcom = intersect(Ic{Fr(icom)},Ic{Fr(jcom)}) ;
            Si ~isempty(zcom) ;
                Acom(Fr(icom),Fr(jcom)) = zcom(1) ;
                %Cas d'un deuxieme sommet commun
                Si length(zcom)==2
                    Bcom(Fr(icom),Fr(jcom)) = zcom(2) ;
                Fin ;
            Fin ;
        Fin ;
    Fin ;
Fin ;
```

- Moyenne pondérée des normales, cas d'un premier sommet commun :

```
Pour icom = Fr
    Pour jcom = Fr
        Si Acom(icom,jcom)~=0
            R(Acom(icom,jcom)) = ...
                An(icom,jcom)*Rc{icom}(Acom(icom,jcom)) + ...
                An(jcom,icom)*Rc{jcom}(Acom(icom,jcom)) ;
            R(Acom(icom,jcom)) = ...
                R(Acom(icom,jcom))/abs(R(Acom(icom,jcom))) ;
        Fin ;
    Fin ;
Fin ;
```

- Moyenne pondérée des normales, cas d'un second sommet commun :

```
Pour icom = Fr
```

```

    Pour jcom = Fr
        Si Bcom(icom,jcom)~=0
            R(Bcom(icom,jcom)) = ...
                An(icom,jcom)*Rc{icom}(Bcom(icom,jcom)) + ...
                    An(jcom,icom)*Rc{jcom}(Bcom(icom,jcom));
            R(Bcom(icom,jcom)) = ...
                R(Bcom(icom,jcom))/abs(R(Bcom(icom,jcom)));
        Fin ;
    Fin ;
Fin ;
Fin;%du if nargin < 12 | isempty(An)

```

La première fonction de base de E_{i_1} est la $N1 + 1$ -ème fonction de base de l'espace EF produit courant E , avec $N1 + 1$ somme des dimensions des $i_1 - 1$ premiers espaces composant l'espace produit E , ensuite, la position de la seconde fonction de base de E_{i_2} est $N2 + 2$, somme des $i_2 - 1$ premiers espaces composants E :

```
N1 = sum(Dnc(1 :i1-1)) ;
```

```
N2 = sum(Dnc(1 :i2 - 1)) ;
```

- Prise en compte de θ :

```
Si theta~=0
```

```
    R = R*exp(sqrt(-1)*theta) ;
```

```
Fin ;
```

- Calcul de la matrice de passage. Ci-dessous I représente les numéros, dans la numérotation des *ddl* de E_{i_1} , des *ddl* concernés par le changement de base et $R(I)$ la valeur de la normale en ces points. La matrice P de passage de la base canonique de l'espace EF E à la nouvelle base est la suivante, on remarque qu'elle est composée de matrices de rotations :

```
Pour k=1 :length(I)
```

```
    P(N1 + I(k), N1 + I(k)) = cos(angle(R(I(k)))) ;
```

```
    P(N2 + I(k), N2 + I(k)) = cos(angle(R(I(k)))) ;
```

```
    P(N1 + I(k), N2 + I(k)) = -sin(angle(R(I(k)))) ;
```

```
    P(N2 + I(k), N1 + I(k)) = sin(angle(R(I(k)))) ;
```

```
Fin ;
```

```
P2 = P ;
```

Fonction :

```
function [P2, P2INV] = changt_base_a(P, PINV, T, Xc, Yc, Arc,...
                                   Pfc, Dnc, D, ia, Fr, A, varargin) ;
```

Calcul de matrice de changement de base.

La commande

```
[P2, P2INV] = changt_base_a(P, PINV, T, Xc, Yc, Arc,...
                             Pfc, Dnc, D, ia, Fr, A) ;
```

effectue le calcul de matrice de changement de base relatif à la matrice A sur les frontières de numéro Fr et par rapport aux inconnues ia .

Cette fonction est utilisée pour les conditions de type $A\tilde{X}(x) = b(x)$ abordées dans le Tableau 2.2.4-2b du Paragraphe 2.2.4 et le commentaire qui suit ce tableau. Les aspects mathématiques de ce changement de base sont abordés dans l'exemple 3 du Paragraphe 3.2.1 sur les conditions au bord et le lien avec la fonction `changt_base_a` au Paragraphe 3.2.3.

Cette fonction intervient dans l'algorithme général du code comme indiqué au Paragraphe 4.1.2 et dans la fonction `EF_assemblage` comme décrit au Paragraphe 4.3, dans la description de la fonction `EF_assemblage`.

Exemple. Un exemple d'utilisation de cette fonction pour le traitement de conditions $A\tilde{X}(x) = b(x)$ est donné au Paragraphe 2.6.3 pour la résolution du problème $(\mathcal{P}_2'')$ par la commande

```
cd C :/EF_text/exemples/exemple_lap_syst4, instruct1
```

On pourra aussi regarder la résolution du problème $(\mathcal{P}_2''')$ du Paragraphe 2.6.3 avec la commande

```
cd C :/EF_text/exemples/exemple_lap_syst4, instruct1b
```

La fonction `changt_base_a` est en développement et fonctionne. Elle est analogue à la fonction `changt_base_nt`, son algorithme n'est pas détaillée ici, on pourra consulter son `help` ou son *script*.

Fonction :

```
function N = norm_energ_cell(donnees, Uc, varargin) ;
```

On donne les formes d'appel les plus courantes, les autres types d'appel sont décrits dans le **help de la fonction**. Tout d'abord deux formes d'appel qui utilisent le fichier désigné par `donnees.nom_prep` et la valeur de la variable `donnees.D.f` :

a)

```
N = norm_energ_cell(donnees, Uc) ;
```

ou

b)

```
N = norm_energ_cell(donnees, Uc, c, cx, cy) ;
```

Si `Uc` est la représentation sous forme de vecteur cellules d'une fonction u de l'espace EF produit $E = E_1 \cdots \times \cdots E_n$ (cf. Paragraphe 1.5), alors `N` est la norme suivante :

$$\left[\int_{\Omega} \sum_{i=1}^n c(i) u_i^2(x, y) + cx(i) u_{i,x}^2(x, y) + cy(i) u_{i,y}^2(x, y) dx dy \right]^{\frac{1}{2}}$$

avec les valeurs par défaut $c(i) = cx(i) = cy(i) = 1$, dans le premier type d'appel, pour $1 \leq i \leq n$.

Ces deux appels utilisent uniquement les champs `donnees.nom_prep` et `donnees.D.f` de la variable `donnees` (cf. Tableaux 2.2.1-1-a et c du Paragraphe 2.2.1). La variable `SL_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3 est chargée à partir du fichier désigné par `donnees.nom_prep`. On peut récupérer cette variable en sortie par

```
[N, SL_Dom] = norm_energ_cell(donnees, Uc) ;
```

ou

```
[N, SL_Dom] = norm_energ_cell(donnees, Uc, c, cx, cy) ;
```

La commande

```
N = norm_energ_cell(donnees, Uc, c, cx, cy, SL_Dom) ;
```

réalise les mêmes opérations que les commandes a) et b) ci-dessus. Ces commandes n'utilisent que le champ `donnees.D.f` de la variable `donnees`.

Les commandes

c)

```
N = norm_energ_cell(donnees, Uc, c, cx, cy, [], X, Y, T) ;
```

ou

```
N = norm_energ_cell(donnees, Uc, c, cx, cy, [], X, Y, T, ...
                    Xc, Yc, Tc) ;
```

réalisent les mêmes opérations que les commandes a) et b) ci-dessus mais avec un calcul de `SL_Dom`. Dans la deuxième commande les variables `Xc`, `Yc` et `Tc` sont recalculées à partir de `donnees` et de `X`, `Y` et `T`. Les champs de `donnees` utilisés sont `donnees.D.f`, `donnees.D.E`, `donnees.D.intT` et `donnees.der`.

Enfin, la commande d)

```
N = norm_energ_cell(donnees, Uc, c, cx, cy, []);
```

effectue les mêmes opérations que la commande c) ci-dessus mais en chargeant le maillage. Elle utilise aussi les mêmes champs de `donnees` qu'en c) avec, en plus, le champ `donnees.nom_mail` pour le chargement du maillage.

Ci-dessus, on peut aussi poser `c`, `cx` ou `cy` à `[]`. Si `c = []` on aura alors par défaut $c(i) = 1$, pour $1 \leq i \leq n$, *idem* pour les deux autres variables.

Variables d'entrée :

donnees : `STRUCT ARRAY`. Variable décrivant les données du problème EF. Les champs suivant sont éventuellement utilisés (*cf.* leur utilisation ci-dessus).

- `donnees.nom_prep` : `CHAR ARRAY`. Nom du fichier relatif au espaces d'EF et aux formules d'intégration (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

Ce champ permet de récupérer la variable `SL_Dom` du Tableau 2.2.3-1-a du Paragraphe 2.2.3 qui contient les matrices d'assemblage relatif au produit scalaire $[H^1(\Omega)]^n$ sur l'espace EF produit courant E .

- `donnees.nom_mail` : `CHAR ARRAY`. Nom du fichier au maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

Dans les commandes ci-dessus, ce champ n'est utilisé que dans la commande d) ci-dessus qui effectue les mêmes opérations que la commande c) ci-dessus, mais en chargeant le maillage. Voir aussi le `help` de la fonction pour ce cas.

Enfin, *cf.* le Tableau 2.2.1-1-c du Paragraphe 2.2.1 :

- `donnees.D.f`,
- `donnees.D.E`,
- `donnees.D.intT`,
- `donnees.D.der`.

Uc : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

- `varargin{1}`, `varargin{2}`, `varargin{3}` : `DOUBLE ARRAY`. Coefficients `c`, `cx` et `cy` ci-dessus :

`c`, `cx`, `cy` : `DOUBLE ARRAY`. Vecteur de taille `length(Uc)`. Pondération des termes en `Uc{i}` intervenant dans la norme de l'énergie. Dans l'appel

`N = norm_energ_cell(donnees, Uc) ;`

ou

`N = norm_energ_cell(donnees, Uc, [], [], []) ;`

Les variables de pondération prennent la valeur par défaut

`c = cx = cy = ones(1, length(Uc)).`

- `varargin{4}`. `CELL ARRAY`. Terme `SL_Dom` vu dans les appels de fonction ci-dessus :

`SL_Dom` : `CELL ARRAY`. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par `donnees.D.E(k)`. Le terme `SL_Dom{i, j, k, m}` est une matrice de type `SPARSE ARRAY` de dimension $\dim(F_i) \times \dim(F_j)$. La matrice `SL_Dom{i, j, k, m}` est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par

$$\int_{\Omega} v_{,k-1} w_{,m-1}.$$

Le terme (p, q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$, où $\{u^k\}_k$ désigne la base de l'espace d'EF F_k . Les bases sont décrites au Paragraphe 1.4 et au Paragraphe 2.3.2, sur un maillage exemple. Cette variable représente le champ `SL_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

- `varargin{5}`, `varargin{6}`. Termes `X` et `Y` de maillage vus dans les appels de fonction ci-dessus :

`X, Y` : `DOUBLE ARRAY`. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{7}`. Termes `T` de maillage vu dans les appels de fonction ci-dessus :
`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{8}`, `varargin{9}`. Termes `Xc` et `Yc` de *ddl* vus dans les appels de fonction ci-dessus :

`Xc`, `Yc` : `CELL ARRAY`. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type `DOUBLE ARRAY`. Les vecteurs `Xc{i}` et `Yc{i}` sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

- `varargin{10}`. Termes `Tc` de *ddl* vu dans les appels de fonction ci-dessus :
`Tc` : `CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

N : DOUBLE ARRAY. Scalaire égal à la norme de type l'énergie calculée.

varargout{1}. Variable SL_Dom décrite ci-dessus.

Exemple : L'exemple suivant est donné par le fichier *script exemple_norm_energ_cell.m* du répertoire **Assemblage** (cf. aussi le **help** de **norm_energ_cell**). Il porte sur le maillage de la Figure 4.3-5.

L'espace EF produit considéré est l'espace $E = P_3^c \times P_3^c \times P_1^c$ avec les commandes :

```
%Choix des types d'espaces : P1 et P3 :
```

```
D.E = [1,3] ;
```

```
%Choix de l'espace produit P3 x P3 x P1 :
```

```
D.f = [2,2,1]
```

Cet exemple effectue d'abord le calcul de la norme $[H^1(\Omega)]^3$ (qui est aussi la norme $[L^2(\Omega)]^3$) de la fonction nulle. Puis avec 7 types d'appels différents de la fonction **norm_energ_cell**, la norme $[H^1(\Omega)]^3$ de la fonction $u(x, y) = (0, 0, 1)$.

Ensuite, cet exemple effectue le calcul de la norme $[L^2(\Omega)]^3$ de la fonction $u(x, y) = (0, 0, x)$ et de sa dérivée $u_{,x}$, on pose alors

$$c = [0, 0, 0], \quad cx = [0, 0, 1], \quad cy = [0, 0, 0].$$

Ensuite, pour la fonction $u(x, y) = x^2$, en effectuant les opérations sur la fonction $\tilde{u}(x, y) = (x^2, 0, 0)$, cet exemple calcule la norme $L^2(\Omega)$ de l'interpolé P_1 aux *ddl* de l'espace d'EF de u et de la dérivée par rapport à x de cet interpolé. L'erreur d'interpolation donne une erreur d'intégration sur la fonction u .

L'exemple effectue les mêmes calculs mais avec une interpolation P_3 , pour ceci on utilise la fonction $\tilde{u}(x, y) = (0, 0, x^2)$, ceci donne un calcul exact, ce qui est dû au fait que u et son interpolé coïncident.

Enfin, on termine avec le calcul de la norme $L^2(\Omega)$ de l'interpolé P_3 de la fonction $x^2 + y^2$ (qui vaut $x^2 + y^2$).

EF/exemples/Assemblage/exemple_norm_energ_cell.m

```
%Fichier EF/exemples/Assemblage/exemple_norm_energ_cell.m
```

```
% Utilisation de norm_energ_cell : calcul de la norme d'un
```

```
% element de l'espace d'EF : ici E = P3 x P3 x P1 continu.
```

```
%
```

```
%Commande :
```

```
%
```

```

%   cd C :/EF/exemples/Assemblage, exemple_norm_energ_cell

clear
disp('Integration sur le carre unite de diverses fonctions');
%Choix des types d'espaces : P1 et P3 :
D.E = [1,3];

%Choix de l'espace produit P3 x P3 x P1 :
D.f = [2,2,1];

%Initialisation des variables T, X, Y du maillage :
T = [1,5,4; 1,2,5; 2,6,5; 2,3,6; 4,8,7; 4,5,8; 5,9,8; 5,6,9];
X([1,2,3,4,5,6,7,8,9]) = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1];
Y([1,2,3,4,5,6,7,8,9]) = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1];
Noeu = length(X);
%et de variable frontiere Front du maillage :
Front{1,1} = [7,8,9];
Front{2,1} = [9,6,3];
Front{3,1} = [3,2,1];
Front{4,1} = [1,4,7];

%Choix des formules d'integration :
D.intT = 10;
D.intA = 6;

%Derivations :
D.der = [0,1,2];

%initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D;

donnees.nom_mail = ...
    'C : \EF\exemples\Assemblage\fich_exemple_norm_energ_cell\f_mail';
donnees.nom_prep = ...

```

```
'C : \EF\exemples\Assemblage\fich_exemple_norm_energ_cell\f_prep';

%creation du fichier de maillage
save(donnees.nom_mail, 'T', 'X', 'Y', 'Front', 'Noeu')

%Calcul du fichier relatif au espaces d'EF et aux formules
%d'integration
EF_Prep(donnees);

%Exemples d'utilisation de la fonction norm_energ_cell

%Chargement des coordonnees des DDL :
load(donnees.nom_prep, 'Xc', 'Yc');

%Initialisation de Uc au vecteur representant la fonction nulle de
%l'espace E :
Uc = init_transf_cell('init', [], donnees);

%Calcul de la norme de Uc :
disp('norme H1 de u(x,y) = (0,0,0) : vaut 0');
N1 = norm_energ_cell(donnees, Uc);
disp(N1);

%Initialisation de la troisieme de Uc au vecteur constant egal a 1
%qui correspond a la fonction u = (0,0,1) de E :
Uc{3} = ones(length(Uc{3}),1);

%Calcul de la norme de Uc de 7 manieres differentes :
disp('Norme H1 de u(x,y) = (0,0,1), 7 types d'appels : vaut 1');
format compact
%1) On n'utilise dans donnees uniquement le fichier nomme par
%   donnees.nom_prep et donnees.D.f :
N2 = norm_energ_cell(donnees, Uc);
disp(N2);
```

```

%2) Comme au 1) avec c, cx, cy en entree par default :
[N2p, SL_Dom] = norm_energ_cell(donnees, Uc, [], [], []);
disp(N2p);

%3) On n'utilise dans donnees que donnees.D.f,
%   (donc sans utiliser ni le fichier nomme par donnees.nom_prep
%   ni le fichier donnees.nom_mail), pas de calcul de SL_Dom :
N2p = norm_energ_cell(donnees, Uc, [], [], [], SL_Dom);
disp(N2p);

%4) On n'utilise dans donnees que donnees.D.f, donnees.D.E,
%   donnees.D.intT donnees.D.der et le fichier nomme par
%   donnees.nom_mail :
N2p = norm_energ_cell(donnees, Uc, [], [], [], []);
disp(N2p);

%5) Comme au 3) X,Y,T ne sont pas utilisees
N2p = norm_energ_cell(donnees, Uc, [], [], [], SL_Dom, X, Y, T);
disp(N2p);

%6) On n'utilise dans donnees que donnees.D.f, donnees.D.E,
%   donnees.D.intT, donnees.D.der :
N2p = norm_energ_cell(donnees, Uc, [], [], [], [], X, Y, T);
disp(N2p);

%7) On n'utilise dans donnees que donnees.D.f, donnees.D.E,
%   donnees.D.intT, donnees.D.der, pas de calcul de Xc, Yc, Tc :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []);
N2p = norm_energ_cell(donnees, Uc, [], [], [], [], X, Y, T, ...
                    Xc, Yc, Tc);
disp(N2p);
format
%Uc{3} est initialise a Xc{1} et Uc represente la fonction

```


%u = (0,0,x) de E :

Uc{3} = Xc{1}' ;

%Calcul de norme pour Uc :

disp('Norme L2 de u(x,y) = (0,0,x) : vaut sqrt(1/3) ~ 0.5774') ;

N3 = norm_energ_cell(donnees, Uc, [0,0,1], [0,0,0], [0,0,0]) ;

disp(N3) ;

%Calcul de norme pour Uc :

disp('Norme L2 de u,x pour u(x,y) = (0,0,x) : vaut 1') ;

N4 = norm_energ_cell(donnees, Uc, [0,0,0], [0,0,1], [0,0,0]) ;

disp(N4) ;

%Uc{3} est initialise a Xc{1}.^2 et Uc represente la fonction

%de E qui interpole aux DDL la fonction (0,0,x^2). Ici on a

%seulement interpolation car x^2 n'est pas dans E3 qui est

%seulement P1 continu

Uc{3} = Xc{1}.^2' ;

%Calcul de norme pour Uc :

disp('Norme L2 de l"interpole P1 de u(x,y) = x^2 : la norme L2')

disp('de x^2 est sqrt(1/5) = 0.4472, celle de l''interpole vaut') ;

N5 = norm_energ_cell(donnees, Uc, [0,0,1], [0,0,0], [0,0,0]) ;

disp(N5) ;

%Calcul de norme pour Uc :

disp('Norme L2 de u,x avec u interpole P1 de u(x,y) = x^2 :')

disp('la norme de u,x vaut sqrt(4/3) ~ 1.1547 celle de ') ;

disp('l''interpole vaut') ;

N6 = norm_energ_cell(donnees, Uc, [0,0,0], [0,0,1], [0,0,0]) ;

disp(N6) ;

%Uc{1} est initialise a Xc{1}.^2 et Uc represente la fonction

%u = (x^2,0,0) de E (par d'erreur d'interpolation) :

```

Uc{1} = Xc{2}.^2' ;

%Calcul de norme pour Uc :
disp(['Norme L2 de l"interpole P3 de x^2 : vaut sqrt(1/5)', ...
      ' ~ 0.4472']);
N7 = norm_energ_cell(donnees, Uc, [1,0,0], [0,0,0], [0,0,0]);
disp(N7) ;

%Calcul de norme pour Uc :
disp('Norme L2 de u,x avec u interpole P3 de u(x,y) = x^2 :')
disp('vaut sqrt(4/3) ~ 1.1547') ;
N8 = norm_energ_cell(donnees, Uc, [0,0,0], [1,0,0], [0,0,0]);
disp(N8) ;

%Calcul de norme pour Uc :
Uc{1} = Xc{2}.^2' + Yc{2}.^3' ;
disp('Norme L2 de l"interpole P3 de x^2 + y^3 :')
disp('vaut sqrt(214/420) ~ 0.7138') ;
N9 = norm_energ_cell(donnees, Uc, [1,0,0], [0,0,0], [0,0,0]);
disp(N9) ;

```

Algorithme :

EF/Assemblage/norm_energ_cell.m

On remet ici la partie déclarative de la fonction :

```
function [N,varargout] = norm_energ_cell(donnees, Uc, varargin) ;
```

• Initialisation :

```
N=0 ;
```

Si varargin est vide ou si varargin{1}, varargin{2}, varargin{3} sont égales à [] alors

```
    c = ones(1,length(Uc)) ;
```

```
    cx = c ;
```

```
    cy = c ;
```

sinon

```
    c = varargin{1} ;
```

```

Si isempty(c)
    c = ones(1,length(Uc)) ;
Fin ;
cx = varargin{2} ;
Si isempty(cx)
    cx = ones(1,length(Uc)) ;
Fin ;
cy = varargin{3} ;
Si isempty(cy)
    cy = ones(1,length(Uc)) ;
Fin ;

```

Fin.

- Chargement de la variable SL_Dom (*cf.* Tableau 2.2.3-1a, Paragraphe 2.2.3) :

Si length(varargin) <= 3 :

```
load(donnees.nom_prep,'SL_Dom') ;
```

Si length(varargin) >= 4 :

```
SL_Dom = varargin{4} ;
```

- Si SL_Dom est vide, cette variable est alors calculée en utilisant le fichier désigné par `donnees.nom_mail` si `length(varargin) = 4` ou bien avec les variables X, Y, T si `length(varargin) = 7`. Le calcul de Xc, Yc, Tc est effectué pour cette valeur de `length(varargin)` mais pas pour `length(varargin) = 10`. Le calcul de SL_Dom se fait avec une boucle identique à celle utilisée dans la fonction `matrice_elem_prepa.T` décrite ci-après. Le principe est le suivant.

- Initialisation de SL_Dom.

- Initialisation de f_Dom et Ass_Dom :

```
f_Dom = cell(length(D.E),length(D.E)) ;
```

```
Ass_Dom = [] ;
```

- Test sur le champ `der` qui doit valoir $[0, 1, 2]$. Par exemple si ce champ vaut `[]` les termes de SL_Dom seront tous nuls.

- Boucle de calcul de SL_Dom :

```
Pour i = 1 :size(T,1)
```

```
    [Ass_Dom, SL_Dom, f_Dom] = ...
```

```
        matrice_elem_prepa_T(donnees, i, T, ...
```

```
            Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;
```

```
Fin ;
```

Ensuite, dans la boucle qui suit, on calcule pour chaque valeur de i et de $d = 0, 1, 2$ l'intégrale

$$\int_{\Omega} u_{i,d}^2.$$

Cette intégrale est donnée par

$$\text{sum}(\text{SL\_Dom}\{j, j, d+1, d+1\} * \text{Uc}\{i\} * \text{Uc}\{i\})$$

avec

$$j = \text{donnees.D.f}(i).$$

Le terme j indique que E_i , le i -ème espace de l'espace EF produit E est l'espace d'EF numéro $\text{donnees.D.E}(j)$ et que les informations de SL_Dom relatives à cet espace sont dans $\text{SL\_Dom}\{j, j, :, :\}$ (*cf.* les espace F_j définis au Paragraphe 2.2 et le Tableau 2.2.1-1c, Paragraphe 2.2.1, pour la description des champs D.E et D.f de la variable *donnees*).

Pour $i = 1$ à n , la dimension de l'espace EF E , on effectue :

```
j = donnees.D.f(i) ;
N = N + c(i)*sum(SL_Dom{j,j,1,1}*Uc{i}.*Uc{i}) + ...
      cx(i)*sum(SL_Dom{j,j,2,2}*Uc{i}.*Uc{i}) + ...
      cy(i)*sum(SL_Dom{j,j,3,3}*Uc{i}.*Uc{i}) ;
```

Fin ;

Et on termine par :

```
N = sqrt(N) ;
varargout{1} = SL_Dom ;
```

Fonction :

```
function N = form_bilin_cell(donnees, Ucc, varargin) ;
```

Fonction utilisateur. Calcul de forme bilinéaire. En développement fonctionnelle. On donne ci-dessous une description succincte.

Soient U_c et V_c la représentation sous forme de vecteur cellules d'une fonction u et d'une fonction v de l'espace EF produit $E = E_1 \cdots \times \cdots E_n$, alors l'appel

a)

```
N = form_bilin_cell(donnees, {Uc,Vc}, c) ;
```

calcule la valeur de N donnée par

$$\sum_{0 \leq k, m \leq 2} \sum_{1 \leq i, j \leq n} \int_{\Omega} c\{k, m\}(i, j) u_{i, k}(x, y) v_{j, m}(x, y).$$

La variable c est facultative ; Si cette variable est omise comme dans l'appel :

b)

```
N = form_bilin_cell(donnees, {Uc,Vc}) ;
```

ou si cette variable est posée à $[]$ dans a), alors la valeur de N calculée est

$$\sum_{1 \leq i, j \leq n} \int_{\Omega} u_{i, k}(x, y) v_{j, m}(x, y).$$

Ces deux appels utilisent uniquement les champs `donnees.nom_prep` et `donnees.D.f` de la variable `donnees` (cf. Tableaux 2.2.1-1-a et c du Paragraphe 2.2.1). Dans ces deux appels, la variable `SL_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3 est chargée à partir du fichier désigné par `donnees.nom_prep`. On peut récupérer cette variable en sortie par

```
[N,SL_Dom] = form_bilin_cell(donnees, {Uc,Vc}) ;
```

ou

```
[N,SL_Dom] = form_bilin_cell(donnees, {Uc,Vc}, c) ;
```

Les commandes

```
N = form_bilin_cell(donnees, {Uc,Vc}, [], SL_Dom) ;
```

ou

```
N = form_bilin_cell(donnees, {Uc,Vc}, c, SL_Dom) ;
```

réalisent les mêmes opérations que la commandes a). Ces commandes n'utilisent alors que le champ `donnees.D.f` de la variable `donnees`.

Les commandes c)

```
N = form_bilin_cell(donnees, {Uc,Vc}, c, [], X, Y, T) ;
```

ou

```
N = form_bilin_cell(donnees, {Uc,Vc}, c, [], X, Y, T, ...
                    Xc, Yc, Tc) ;
```

réalisent les mêmes opérations que les commandes a) et b) ci-dessus mais avec un calcul de `SL_Dom`. Dans la deuxième commande les variables `Xc`, `Yc` et `Tc` sont recalculées à partir de `donnees` et de `X`, `Y` et `T`. Les champs de `donnees` utilisés sont `donnees.D.f`, `donnees.D.E`, `donnees.D.intT` et `donnees.der`. Enfin, la commande d)

```
N = form_bilin_cell(donnees, {Uc,Vc}, c, []) ;
```

effectue les mêmes opérations que la commande c) ci-dessus mais en chargeant le maillage. Elle utilise aussi les mêmes champs de `donnees` qu'en c) avec, en plus, le champ `donnees.nom_mail` pour le chargement du maillage.

L'opération

```
N = form_bilin_cell(donnees, {Uc,Uc}, ...)
```

est aussi effectuée par l'appel

```
N = form_bilin_cell(donnees, Uc, ...)
```

Variables d'entrée :

donnees : `STRUCT ARRAY`. Variable décrivant les données du problème EF. Les champs suivant sont éventuellement utilisés (*cf.* leur utilisation ci-dessus)

- **donnees.nom_prep** : `CHAR ARRAY`. Nom du fichier relatif au espaces d'EF et aux formules d'intégration (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

Ce champ permet de récupérer la variable `SL_Dom` du Tableau 2.2.3-1-a du Paragraphe 2.2.3 qui contient les matrices d'assemblage relatif au produit scalaire $[H^1(\Omega)]^n$ sur l'espace EF produit courant E .

- **donnees.nom_mail** : `CHAR ARRAY`. Nom du fichier au maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1).

Dans les commandes ci-dessus, ce champ n'est utilisé que dans la commande d) ci-dessus qui effectue les mêmes opérations que la commande c) ci-dessus, mais en chargeant le maillage. Voir aussi le `help` de la fonction pour ce cas.

Enfin, *cf.* le Tableau 2.2.1-1-c du Paragraphe 2.2.1 :

- **donnees.D.f**,
- **donnees.D.E**,
- **donnees.D.intT**,
- **donnees.D.der**.

Ucc. `CELL ARRAY`. De la forme $U_{cc} = \{U_c, V_c\}$ où

Uc : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de

l'espace EF produit courant E (cf. Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux ddl correspondants sont identiques.

Vc. Variable du même type que **Uc**.

varargin : DOUBLE ARRAY.

- **varargin{1}** :

c : CELL ARRAY. Coefficient de pondération dans le calcul de la forme bilinéaire. Si **varargin** est vide ou si **varargin{1}** est vide, par défaut **c{k,k}** vaut **eye(length(Uc))**, matrice identité de $\mathbb{R}^n$ avec n longueur de **Uc** et **c{k,m}** = [] si **k** est différent de **m**.

Ainsi, dans l'appel :

```
N = form_bilin_cell(donnees, {Uc,Vc}) ;
```

ou l'appel

```
N = form_bilin_cell(donnees, {Uc,Vc}, []) ;
```

Les variables de pondération prennent la valeur par défaut ci-dessus.

- **varargin{2}**. CELL ARRAY. Terme **SL_Dom** vu dans les appels de fonction ci-dessus :

SL_Dom : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par **donnees.D.E(k)**. Le terme **SL_Dom{i,j,k,m}** est une matrice de type **SPARSE ARRAY** de dimension $\dim(F_i) \times \dim(F_j)$. La matrice **SL_Dom{i,j,k,m}** est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par

$$\int_{\Omega} v_{,k-1} w_{,m-1}.$$

Le terme (p,q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$, où $\{u^k\}_k$ désigne la base de l'espace d'EF F_k . Les bases sont décrites au Paragraphe 1.4 et au Paragraphe 2.3.2, sur un maillage exemple. Cette variable représente le champ **SL_Dom** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

- **varargin{3}**, **varargin{4}**. Termes **X** et **Y** de maillage vus dans les appels de fonction ci-dessus :

X,Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- **varargin{5}**. Termes **T** de maillage vu dans les appels de fonction ci-dessus :

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{6}`, `varargin{7}`. Termes `Xc` et `Yc` de *ddl* vus dans les appels de fonction ci-dessus :

`Xc`, `Yc` : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs `Xc{i}` et `Yc{i}` sont les vecteurs des coordonnées en *x* et *y* des *ddl* du *i*-ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

- `varargin{8}`. Termes `Tc` de *ddl* vu dans les appels de fonction ci-dessus :

`Tc` : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du *i*-ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`N` : DOUBLE ARRAY. Scalaire égal à la norme de type l'énergie calculée.

`varargout{1}`. Variable `SL_Dom` décrite ci-dessus.

Exemple.

Un exemple d'utilisation est donné par la commande :

```
cd C :/EF/exemples/Assemblage, exemple_form_bilin_cell
```

Algorithme. La déclaration de la fonction est

```
function N = form_bilin_cell(donnees, Ucc, varargin) ;
```

L'algorithme suit le principe de la fonction `norm_energ_cell` pour la gestion des variables. La boucle de calcul de `N` est donnée par

```
for k=1 :3
    for m=1 :3
        for i=1 :size(c{k,m},1)
            for j=1 :size(c{k,m},2)
                pi = donnees.D.f(i) ;
                qj = donnees.D.f(j) ;
                N = N + ...
                    c{k,m}(j,i)*sum(SL_Dom{pi,qj,k,m}*U{j}.*V{i}) ;
            end ;
        end ;
    end ;
end ;
```


Nous rappelons ci-dessous la description des fonctions de calcul de norme : `norm_T`, `norm_A` et `norm_Ad2` dont l'utilisation est décrite au Paragraphe 2.5 consacré au traitement des solutions. Nous donnons aussi une description de la fonction d'intégration `int_fonc_T` qui est d'emploi analogue à la fonction `norm_T`.

Les exemples d'utilisation de ces 4 fonctions portent sur un des problèmes de transport étudiés au Paragraphe 2.7 qui est le suivant :

$$(\mathcal{P}_3) \begin{cases} \text{Il existe } u \text{ tel que :} \\ u + \frac{\partial u}{\partial \beta} = 0 \text{ sur } \Omega \\ u = 1 \text{ sur } \Gamma_{\beta}^{-}, \end{cases}$$

où le domaine Ω de frontière Γ est le carré unité, β désigne le champ de vecteurs $\beta = (1, 0)$, on pose

$$\frac{\partial u}{\partial \beta} = \frac{\partial u}{\partial x},$$

enfin $\Gamma_{\beta}^{-} = \{(0, y) \in \Gamma\}$ est la partie de la frontière par où le champ $\beta = (1, 0)$ rentre. Cette frontière sera représenté par le morceau de frontière numéro 4 dans la structure de donnée du maillage. La solution exacte de ce problème est donnée par

$$u(x, y) = e^{-x}.$$

Dans les exemples ci-dessous le problème est résolu avec du P_1 continu avec la méthode avec décentrage de type SUPG (paramètre de décentrage 0.1) par la commande :

```
cd C :/EF_text/exemples/exemple_transport, instruct1_supg
```

Fonction :

```
function S = norm_T(donnees, Uc, nomf, varargin) ;
```

Fonction utilisateur (*cf.* Paragraphe 2.5, partie calcul de normes). Calcul de norme $L^p(\Omega)$ sur le domaine Ω .

La commande

```
S = norm_T(donnees, Uc, nomf) ;
```

calcule la norme $L^2(\Omega)$ de la fonction désignée par `nomf`, avec `Uc` représentant un élément de l'espace EF produit E défini par `donnees.D.E` et `donnees.D.f`.

La commande

```
S = norm_T(donnees, Uc, nomf, p) ;
```

calcule la norme $L^p(\Omega)$ avec $1 \leq p \leq \infty$ et la norme $L^2(\Omega)$ si `p = []`.

Par rapport aux commandes ci-dessus, les commandes suivantes n'utilisent pas les fichiers `donnees.nom_mail` et `donnees.nom_prep` :

```
S = norm_T(donnees, Uc, nomf, P, X, Y, T) ;
```

et, sans recalcul des degrés de liberté :

```
S = norm_T(donnees, Uc, nomf, P, X, Y, T, Tc) ;
```

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF. Pour les appels du type :

```
S = norm_T(donnees, Uc, nomf) ;
S = norm_T(donnees, Uc, nomf, P) ;
```

les champs de la variable **donnees** à initialiser sont

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der, ([0,1,2])
donnees.nom_prep,
donnees.nom_mail.
```

Et pour les appels du type :

```
S = norm_T(donnees, Uc, nomf, P, X, Y, T) ;
S = norm_T(donnees, Uc, nomf, P, X, Y, T, Tc) ;
```

avec appel des variables de maillage X, Y, T et éventuellement de Tc, les champs de la variable **donnees** à initialiser sont :

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der.
```

Dans ce cas la fonction **norm_T** n'utilise pas le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) qui contiennent des variables de maillage X, Y, T et de *ddl* dont la variable Tc.

Uc : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant *E* (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

nomf : CHAR ARRAY ou FUNCTION_HANDLE ARRAY. Handle sur la fonction dont on veut calculer la norme. Syntaxe :

```
Rc = f(donnees, uc, a1, b1, c1, a2, ...
      b2, c2, de, Ac, Bc, numK) ;
```

En entrée, la variable **donnees** est la variable de type **struct** décrivant le problème, **uc** la valeur de l'itéré courant et de ses dérivées aux points d'intégration sur le

triangle courant K de numéro `numK`. Ensuite `a1`, `b1`, `c1`, `a2`, `b2` et `c2` désignent les coordonnées cartésiennes des sommets de K (les trois sommets ont alors pour coordonnées $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$), `Ac` et `Bc` les coordonnées cartésiennes, en x et en y , des points d'intégration. La variable de sortie `Rc` est la variable de de type `cell` décrivant la valeur aux points de la formule de quadrature de la fonction à intégrer.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme : `varargin`. Si `varargin` est vide alors le terme `p` dans les appels ci-dessus pour le calcul en norme L^p , vaut `p = 2`.

- `varargin{1}`. Si `varargin` est vide alors `p = 2`, sinon `p = varargin{1}` :

`P : DOUBLE ARRAY`. désigne p dans le calcul de la norme $L^p(\Omega)$.

- `varargin{2}`, `varargin{3}`. Termes `X` et `Y` de maillage vus dans les appels de fonction ci-dessus :

`X, Y : DOUBLE ARRAY`. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{5}`. Terme `T` de maillage vu dans les appels de fonction ci-dessus :

`T : DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{3}`. Terme `Tc` de `ddl` vu dans les appels de fonction ci-dessus :

`Tc : CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les `ddl` du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`S : DOUBLE ARRAY`. Valeur de la norme $L^p(\Omega)$ de la fonction designée par `nomf`.

Exemple. L'exemple suivant est tiré du `help` de la fonction `norm_T` :

EXEMPLE 1

```
%Ci-dessous f1 est la fonction erreur sur la solution exacte
%S1 = S2 = S3 = 2.8559e-004
cd C :/EF_text/exemples/exemple_transport, instruct1_supg,
cd C :/EF_text/exemples/exemple_transport/equ
f1 = @eq_norm_Dom ;
S1 = norm_T(donnees, Uc, f1) ;
disp('erreur') ;
```

```

disp(S1) ;
load(donnees.nom_mail, 'X', 'Y', 'T') ;
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []) ;
S2 = norm_T(donnees, Uc, f1, [], X, Y, T) ;
disp('erreur') ;
disp(S2) ;
load(donnees.nom_prep, 'Tc') ;
S3 = norm_T(donnees, Uc, f1, [], X, Y, T, Tc) ;
disp('erreur') ;
disp(S3) ;

```

Dans cet exemple, la fonction de calcul de norme de `handle` de fonction `f1 = @eq_norm_Dom` est donnée par le fichier *script* suivant.

C :/EF_text/exemples/exemple_transport/equ/eq_norm_Dom (Exemple norm_T)

```

function Rc = eq_norm_Dom(donnees, uc, a1, b1, c1, a2, ...
                          b2, c2, de, Ac, Bc, numK) ;
Rc{1} = exp(-Ac) - uc{1,1} ;

```

Cette fonction permet alors le calcul de l'erreur

$$\left[\int_{\Omega} (u(x,y) - e^{-x})^2 \right]^{1/2},$$

la fonction à intégrer étant ici

$$f(x,y) = u(x,y) - e^{-x}$$

où u est la solution du problème EF.

Dans la fonction `eq_norm_Dom`, les valeurs de u et de ses dérivées d'ordre 1 aux points de la formule de quadrature de coordonnées cartésiennes `Ac` et `Bc` sur le triangle de numéro `numK`, sont données par le tableau de cellules `uc` (cf la description de cette variable au Tableau 2.2.4-1 du Paragraphe 2.2.4, sur la description des fonctions définissant les EDP).

Algorithme :

EF/Assemblage_elementaire/norm_T.m

On remet ici la partie déclarative de la fonction :

```
function S = norm_T(donnees, Uc, nomf, varargin) ;
```

On charge ou on calcule les variables X, Y, T, Tc.

P vaut 2 par défaut.

```
Pour i=1 :size(T,1)
```

Cette partie est identique à la partie de la fonction `matrice_elem_T`.

- Ci-dessous `ptc{ic}` contient les numéros de *ddl* du *ic*-ième type d'espace qui appartiennent au triangle numéro *i* :

```
Pour ic=1 :length(D.E)
    ptc{ic} = Tc{ic}(i, :) ;
Fin ;
```

- Coordonnées des 3 sommets du triangle numéro *i* que l'on note pour la suite *a*, *b* et *c* de coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) :

```
XT = X(T(i,1 :3)) ;
YT = Y(T(i,1 :3)) ;
```

- Valeur des composantes de la solution associées aux *ddl* :

%Boucle sur les espaces de l'espace produit

```
Pour ic=1 :length(D.f)
    Uc_pt{ic} = Uc{ic}(ptc{D.f(ic)}) ;
Fin ;
```

- Calculs de coefficients pour réaliser l'assemblage sur le triangle numéro *i*. Ces coefficients sont :

- Coordonnées respectives des 3 sommets du triangle : $(a1, a2) = (a_1, a_2)$, $(b1, b2) = (b_1, b_2)$, $(c1, c2) = (c_1, c_2)$,

- Valeur du Jacobien : $de = (a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'$,

- Coordonnées cartésiennes des points de la formule de quadrature : *Ac* et *Bc*,

- Coefficients associés au points de la formule de quadrature : *coeff*,

- Valeurs au points de la formule de quadrature des coordonnées barycentriques relatives à *a*, *b*, *c* : *L1*, *L2*, *L3*

```
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] ...
    = assemble_coeff_T(XT, YT, D) ;
```

a) Calcul par type d'espace de numéro `D.E{ic}` des coefficients d'intégration sous forme dupliquée à la dimension de l'espace :

- Boucle sur les types d'espaces utilisés

```
Pour ic=1 :length(D.E)
    I{ic} = pre_assemble_T([], D.E(ic), [], a1, b1, c1, ...
        a2, b2, c2, de, Ac, Bc, coeff, 1, D, L1, L2, L3);
Fin;
```

Fin de la partie identique à la partie de la fonction `matrice_elem_T`.

- Calcul de la norme.

```
Rc = feval(nomf,donnees, uc, a1, b1, c1, a2, ...
    b2, c2, de, Ac, Bc, i);

Si p < Inf
    Pour ia=1 :length(Rc)
        S = S + sum( (abs(Rc{ia}).^p).*coeff);
    Fin;
Fin;

Si p==Inf
    Pour ia=1 :length(Rc)
        S = max(S,max(abs(Rc{ia})));
    Fin;
Fin;

Fin;

Si p < Inf
    %Valeur absolue car on peut avoir S < 0 a
    % cause de l'erreur d'integration :
    S = abs(S)^(1/p);
Fin;%du Pour i=1 :size(T,1)
```

Fonction :

```
function S = norm_A(donnees, Uc, nomf, varargin) ;
```

Fonction utilisateur (*cf.* Paragraphe 2.5, partie calcul de normes). Calcul de norme $L^p(\Gamma)$ sur la frontière Γ du domaine.

La commande

```
S = norm_A(donnees, Uc, nomf) ;
```

calcule la norme $L^2(\Gamma)$ de la fonction désignée par **nomf**, avec **Uc** représentant un élément de l'espace EF produit E défini par **donnees.D.E** et **donnees.D.f**.

La commande

```
S = norm_A(donnees, Uc, nomf, p) ;
```

calcule la norme $L^p(\Gamma)$ avec $1 \leq p \leq \infty$ et la norme $L^2(\Gamma)$ si **p** = [].

Par rapport aux commandes ci-dessus, les commandes suivantes n'utilisent pas les fichiers **donnees.nom_mail** et **donnees.nom_prep** :

```
S = norm_A(donnees, Uc, nomf, p, X, Y, T, Front) ;
```

et, sans recalcul des degrés de liberté :

```
S = norm_A(donnees, Uc, nomf, p, X, Y, T, Front, Tc) ;
```

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF. Pour les appels du type :

```
S = norm_A(donnees, Uc, nomf) ;
```

```
S = norm_A(donnees, Uc, nomf, P) ;
```

les champs de la variable **donnees** à initialiser sont

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der, ([0,1,2])
donnees.nom_prep,
donnees.nom_mail.
```

Et pour les appels du type :

```
S = norm_A(donnees, Uc, nomf, P, X, Y, T) ;
```

```
S = norm_A(donnees, Uc, nomf, P, X, Y, T, Tc) ;
```

avec appel des variables de maillage **X**, **Y**, **T** et éventuellement de **Tc**, les champs de la variable **donnees** à initialiser sont :

```

donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der.

```

Dans ce cas la fonction `norm_A` n'utilise pas le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) qui contiennent des variables de maillage `X`, `Y`, `T` et de `ddl` dont la variable `Tc`.

`Uc` : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux `ddl` correspondants sont identiques.

`nomf` : `CHAR ARRAY` ou `FUNCTION_HANDLE ARRAY`. Handle sur la fonction dont on veut calculer la norme. Syntaxe :

```

Rc = feval(nomf,donnees, numF, uc, a1, b1, c1, ...
           a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;

```

En entrée, la variable `donnees` est la variable de type `struct` décrivant le problème, `numF` le numéro de frontière courante, `uc` la valeur de l'itéré courant et de ses dérivées aux points d'intégration sur les arêtes du triangle numéro `numK` qui porte l'arête courante de numéro global `numA` et de numéro local `numALoc` sur ce triangle. Ensuite, `a1`, `b1`, `c1`, `a2`, `b2` et `c2` désignent les coordonnées cartésiennes des sommets du triangle numéro `numK` (les trois sommets ont alors pour coordonnées $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$ et `Ac` et `Bc` les coordonnées cartésiennes, en x et en y , des points d'intégration.

La variable de sortie `Rc` est la variable de de type `cell` décrivant la valeur aux points de la formule de quadrature de la fonction à intégrer.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme : `varargin`. Si `varargin` est vide alors le terme `p` dans les appels ci-dessus pour le calcul en norme L^p , vaut `p = 2`.

- `varargin{1}`. Si `varargin` est vide alors `p = 2`, sinon `p = varargin{1}` :

`P` : `DOUBLE ARRAY`. désigne p dans le calcul de la norme $L^p(\Omega)$.

- `varargin{2}`, `varargin{3}`. Termes `X` et `Y` de maillage vus dans les appels de fonction ci-dessus :

`X,Y` : `DOUBLE ARRAY`. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{5}`. Terme `T` de maillage vu dans les appels de fonction ci-dessus :
`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{3}`. Terme `Tc` de `ddl` vu dans les appels de fonction ci-dessus :
`Tc` : `CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les `ddl` du *i*-ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`S` : `DOUBLE ARRAY`. Valeur de la norme $L^p(\Gamma)$ de la fonction designée par `nomf`.

Algorithme. Sur le principe de `norm_T` avec cette fois-ci une boucle sur la variable d'arêtes frontières `Pf` (*cf.* le Tableau 2.2.3-1b du Paragraphe 2.2.3) au lieu d'une boucle sur la variable de triangles `T` et sur le principe de la fonction `matrice_elem_A` au lieu de `matrice_elem_T`.

Exemple. L'exemple suivant est tiré du `help` de la fonction `norm_A` :

EXEMPLE 1

```
%Ci-dessous f1 est sur la frontiere la fonction erreur sur la
%solution multipliee par n.U avec n normale exterieure
%scalaire et U le champ qui transporte dans le probleme
%ci-dessous. On a
%S1 = S2 = S3 = 3.9538e-004
cd C :/EF_text/exemples/exemple_transport, instruct1_supg,
cd C :/EF_text/exemples/exemple_transport/equ
f1 = @eq_norm_Fr ;
S1 = norm_A(donnees, Uc, f1) ;
disp('erreur') ;
disp(S1) ;
load(donnees.nom_mail, 'X', 'Y', 'T', 'Front')
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []) ;
S2 = norm_A(donnees, Uc, f1, [], X, Y, T, Front) ;
disp('erreur') ;
disp(S2) ;
S3 = norm_A(donnees, Uc, f1, [], X, Y, T, Front, Tc) ;
disp('erreur') ;
disp(S3) ;
```

Dans cet exemple, la fonction de calcul de norme de `handle` de fonction `f1 = @eq_norm_Fr` est donnée par :

C :/EF_text/exemples/exemple_transport/equ/eq_norm_Fr (exemple norm_A)

```
function Rc = eq_norm_Fr(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA);
%Termes intermediaires :
Zero = zeros(1,length(Ac));
%Champ qui transporte :
f = donnees.champ_u;
[u1,u2] = feval(f,Ac,Bc);
%Valeur de n.u au bord, n normale unitaire exterieure :
if numF==1, nu = u2; end;
if numF==2, nu = u1; end;
if numF==3, nu = -u2; end;
if numF==4, nu = -u1; end;
if numF==0, nu = Zero; end;
%nu ci-dessous vaut |u.n| si u.n <= 0 et vaut 0 sinon :
nu = abs(nu).*(nu <= 0);
Rc{1} = (exp(-Ac) - uc{1,1}).*sqrt(nu);
```

Cette fonction permet alors le calcul de l'erreur

$$\left[\int_{\Gamma_{\beta}^{-}} (u(x,y) - e^{-x})^2 |\beta \cdot n| ds \right]^{1/2},$$

la fonction à intégrer valant ici

$$f(x,y) = u(x,y) - e^{-x}$$

sur Γ_{β}^{-} et vaut 0 sur les autres frontières du carré. On pose aussi $f = 0$ sur Σ la réunion des arêtes intérieures, cette intialisation sur Σ est ici facultative, car `norm_A` n'effectue les calculs que sur la frontière Γ . Dans l'expression de f , u est la solution du problème EF.

Dans la fonction `eq_norm_Fr`, les valeurs de u et de ses dérivées d'ordre 1, aux points de coordonnées cartésiennes `Ac` et `Bc` de la formule de quadrature sur les arêtes du triangle de numéro `numK` portant l'arête frontière courante `numA` de numéro local `numALoc`, sont données par le tableau de cellules `uc` (*cf.* la description de cette variable au Tableau 2.2.4-1 du Paragraphe 2.2.4, sur la description des fonctions définissant les EDP).

Fonction :

```
function S = norm_Ad(donnees, Uc, nomf, varargin) ;
```

Fonction utilisateur (*cf.* Paragraphe 2.5, partie calcul de normes). Calcul de norme $L^p(\Gamma \cup \Sigma_1 \cup \Sigma_2)$ sur la réunion de Γ la frontière du domaine, de Σ_1 et Σ_2 . Les ensembles Σ_1 et Σ_2 sont tous deux égaux à la réunion des arêtes intérieures Σ , l'indice 1 (resp. 2) indique que pour une arête donnée, on intègre sa restriction au premier (resp. second) triangle qui porte cette arête. Pour l'arête i le premier triangle a pour numéro $\text{Pa}(i, 1)$ et le second $\text{Pa}(i, 3)$, $1 \leq i \leq \text{size}(\text{Pa}, 1)$ (*cf.* la variable **Pa** au Tableau 2.2.3-1b).

La commande

```
S = norm_Ad(donnees, Uc, nomf) ;
```

calcule la norme $L^2(\Gamma \cup \Sigma_1 \cup \Sigma_2)$ de la fonction désignée par **nomf**, avec **Uc** représentant un élément de l'espace EF produit E défini par **donnees.D.E** et **donnees.D.f**. Ci-dessus $\Sigma_1 = \Sigma$ (resp. $\Sigma_2 = \Sigma$) est la réunion des arêtes intérieures où l'intègre les restrictions des fonctions au triangle de numéro $\text{Pa}(i, 1)$ (resp. $\text{Pa}(i, 3)$), $1 \leq i \leq \text{size}(\text{Pa}, 1)$ où **Pa** est la variable de description de données relatives aux arêtes décrite au Tableau 2.2.3-1b.

La commande

```
S = norm_Ad(donnees, Uc, nomf, p) ;
```

calcule la norme $L^p(\Gamma \cup \Sigma_1 \cup \Sigma_2)$ avec $1 \leq p \leq \infty$ et la norme $L^2(\Gamma \cup \Sigma_1 \cup \Sigma_2)$ si $p = []$.

Par rapport aux commandes ci-dessus, les commandes suivantes n'utilisent pas les fichiers **donnees.nom_mail** et **donnees.nom_prep** :

```
S = norm_Ad(donnees, Uc, nomf, p, X, Y, T, Front) ;
```

et, sans recalcul des degrés de liberté :

```
S = norm_Ad(donnees, Uc, nomf, p, X, Y, T, Front, Tc) ;
```

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF. Pour les appels du type :

```
S = norm_A(donnees, Uc, nomf) ;
```

```
S = norm_A(donnees, Uc, nomf, P) ;
```

les champs de la variable **donnees** à initialiser sont

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der, ([0,1,2])
```

```

donnees.nom_prep,
donnees.nom_mail.

```

Et pour les appels du type :

```

S = norm_Ad(donnees, Uc, nomf, P, X, Y, T) ;
S = norm_Ad(donnees, Uc, nomf, P, X, Y, T, Tc) ;

```

avec appel des variables de maillage X, Y, T et éventuellement de Tc, les champs de la variable `donnees` à initialiser sont :

```

donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der.

```

Dans ce cas la fonction `norm_Ad` n'utilise pas le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) qui contiennent des variables de maillage X, Y, T et de *ddl* dont la variable Tc.

`Uc` : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

`nomf` : CHAR ARRAY ou FUNCTION_HANDLE ARRAY. Handle sur la fonction dont on veut calculer la norme. Syntaxe :

```

Rc = feval(nomf,donnees, numF, uc, a1, b1, c1, ...
           a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;

```

En entrée, la variable `donnees` est la variable de type `struct` décrivant le problème, `numF` le numéro de frontière courante, `uc` la valeur de l'itéré courant et de ses dérivées aux points d'intégration sur les arêtes du triangle numéro `numK` qui porte l'arête courante de numéro global `numA` et de numéro local `numALoc` sur ce triangle. Ensuite, `a1`, `b1`, `c1`, `a2`, `b2` et `c2` désignent les coordonnées cartésiennes des sommets du triangle numéro `numK` (les trois sommets ont alors pour coordonnées (`a1`, `a2`), (`b1`, `b2`) et (`c1`, `c2`) et `Ac` et `Bc` les coordonnées cartésiennes, en x et en y , des points d'intégration.

La variable de sortie `Rc` est la variable de de type `cell` décrivant la valeur aux points de la formule de quadrature de la fonction à intégrer.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme :

`varargin`. Si `varargin` est vide alors le terme `p` dans les appels ci-dessus pour le calcul en norme L^p , vaut `p = 2`.

- `varargin{1}`. Si `varargin` est vide alors `p = 2`, sinon `p = varargin{1}` :

`P` : DOUBLE ARRAY. désigne `p` dans le calcul de la norme $L^p(\Omega)$.

- `varargin{2}`, `varargin{3}`. Termes `X` et `Y` de maillage vus dans les appels de fonction ci-dessus :

`X,Y` : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{5}`. Terme `T` de maillage vu dans les appels de fonction ci-dessus :

`T` : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{3}`. Terme `Tc` de *ddl* vu dans les appels de fonction ci-dessus :

`Tc` : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du *i*-ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`S` : DOUBLE ARRAY. Valeur de la norme $L^p(\Gamma \cup \Sigma_1 \cup \Sigma_2)$ de la fonction designée par `nomf`.

Algorithme. Sur le principe de `norm_T` avec cette fois-ci une boucle sur la variable d'arêtes `Pa` (*cf.* le Tableau 2.2.3-1b du Paragraphe 2.2.3) au lieu d'une boucle sur la variable de triangles `T` et sur le principe de la fonction `matrice_elem_Ad` au lieu de `matrice_elem_T`.

Exemple. L'exemple suivant est tiré du `help` de la fonction `norm_Ad`. Ci-dessous, le calcul des variables `S1`, `S2` et `S3` illustre le même calcul de norme, mais avec des syntaxes différentes dans l'appel de `norm_Ad`.

EXEMPLE 1

```
%Ci-dessous f1 est sur la frontiere la fonction erreur sur la
%solution multipliee par n.U avec n normale exterieure
%scalaire et U le champ qui transporte dans le probleme
%ci-dessous et vaut 0 ailleurs : on retrouve alors le meme
%resultat que dans l'exemple de NORM_A. On a
%S1 = S2 = S3 = 3.9538e-004.
%On calcule aussi S4 = 0.0031 l'erreur sur les aretes du
%maillage.
cd C :/EF_text/exemples/exemple_transport, instruct1_supg,
```

```

cd C :/EF_text/exemples/exemple_transport/equ
f1 = @eq_norm_Fr
S1 = norm_Ad(donnees, Uc, f1);
disp('erreur');
disp(S1);
load(donnees.nom_mail, 'X', 'Y', 'T', 'Front')
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []);
S2 = norm_Ad(donnees, Uc, f1, [], X, Y, T, Front);
disp('erreur');
disp(S2);
S3 = norm_Ad(donnees, Uc, f1, [], X, Y, T, Front, Tc);
disp('erreur');
disp(S3);
%f2 est le saut de u + u,x sur les aretes du maillage
f2 = @eq_norm_A1
S4 = norm_Ad(donnees, Uc, f2, [], X, Y, T, Front, Tc);
disp('erreur');
disp(S4);r');
disp(S4);

```

Dans cet exemple, la première fonction de calcul de norme de `handle f1 = @eq_norm_Fr` est identique à celle de l'exemple précédent. Comme la fonction à intégrer est nulle sur la réunion Σ des arêtes intérieures, on obtient le même résultat que précédemment.

La seconde fonction de calcul de norme de `handle f2 = @eq_norm_A1` est la suivante

C :/EF_text/exemples/exemple_transport/equ/eq_norm_A1 (exemple norm_Ad)

```

function Rc = eq_norm_A1(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA);

Rc{1} = exp(-Ac) - uc{1,1};

```

Cette fonction permet alors le calcul de l'erreur

$$\left[\int_{\Sigma_1 \cup \Sigma_2} (u(x, y) - e^{-x})^2 ds + \int_{\Gamma} (u(x, y) - e^{-x})^2 ds \right]^{1/2}$$

la fonction à intégrer étant ici

$$f(x, y) = u(x, y) - e^{-x}$$

où u est la solution du problème EF.

Dans la fonction `eq_norm_A1`, le tableau de cellules `uc` donne les valeurs de u et de ses dérivées d'ordre 1 aux points de coordonnées cartésiennes `Ac` et `Bc` de la formule de quadrature sur les arêtes du triangle de numéro `numK` (*cf.* la description de cette variable au Tableau 2.2.4-1 du Paragraphe 2.2.4, sur la description des fonctions définissant les EDP). Ce triangle (il peut y en avoir un second) porte l'arête frontière courante `numA` de numéro local `numALoc`.

Fonction :

```
function S = norm_Ad2(donnees, Uc, nomf, varargin) ;
```

Fonction utilisateur (*cf.* Paragraphe 2.5, partie calcul de normes). Calcul de norme $L^p(\Gamma \cup \Sigma)$ sur la réunion de Γ la frontière du domaine et de Σ la réunion des arêtes intérieures du maillage.

Par rapport à la fonction **norm_Ad** la fonction f désignée par **nomf**, possède une variable d'entrée en plus : la variable **geomT** portant sur la description de triangles. Cette variable peut être utilisé dans l'algorithme de la fonction f pour calculer des sauts à travers Σ .

D'autre part, dans la fonction f , la fonction u est connue de chaque côté de l'arête courante, ce qui permet le calcul des sauts d'une fonction à travers une arête.

La commande

```
S = norm_Ad2(donnees, Uc, nomf) ;
```

calcule la norme $L^2(\Gamma \cup \Sigma)$ de la fonction désignée par **nomf**, avec **Uc** représentant un élément de l'espace EF produit E défini par **donnees.D.E** et **donnees.D.f**.

La commande

```
S = norm_Ad2(donnees, Uc, nomf, p) ;
```

calcule la norme $L^p(\Gamma \cup \Sigma)$ avec $1 \leq p \leq \infty$ et la norme $L^2(\Gamma \cup \Sigma)$ si $p = []$.

Par rapport aux commandes ci-dessus, les commandes suivantes n'utilisent pas les fichiers **donnees.nom_mail** et **donnees.nom_prep** :

```
S = norm_Ad2(donnees, Uc, nomf, p, X, Y, T, Front) ;
```

et, sans recalcul des degrés de liberté :

```
S = norm_Ad2(donnees, Uc, nomf, p, X, Y, T, Front, Tc) ;
```

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF. Pour les appels du type :

```
S = norm_Ad2(donnees, Uc, nomf) ;
```

```
S = norm_Ad2(donnees, Uc, nomf, P) ;
```

les champs de la variable **donnees** à initialiser sont

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der, ([0,1,2])
donnees.nom_prep,
```



```
donnees.nom_mail.
```

Et pour les appels du type :

```
S = norm_A(donnees, Uc, nomf, P, X, Y, T) ;
```

```
S = norm_A(donnees, Uc, nomf, P, X, Y, T, Tc) ;
```

avec appel des variables de maillage X, Y, T et éventuellement de Tc , les champs de la variable **donnees** à initialiser sont :

```
donnees.D.E,
donnees.D.f,
donnees.D.intT,
donnees.D.der.
```

Dans ce cas la fonction **norm.T** n'utilise pas le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) qui contiennent des variables de maillage X, Y, T et de *ddl* dont la variable Tc .

Uc : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

nomf : CHAR ARRAY ou FUNCTION_HANDLE ARRAY. Handle sur la fonction dont on veut calculer la norme. Syntaxe :

```
function [Rc] = f(donnees, numF, uc, a1, b1, c1, ...
a2, b2, c2, de, Ac, Bc, numK, numALoc, numA, geomT) ;
```

En entrée, la variable **donnees** est la variable de type **struct** décrivant le problème. La variable **numF** désigne le numéro de la frontière courante, ou bien une arête intérieure avec alors le numéro 0.

Les vecteurs **Ac** et **Bc** sont les coordonnées des points d'intégration situés sur l'arête numéro **numA** uniquement, et non ceux de la frontière du (ou des) triangles qui portent cette arête.

Si **geomT.p_A** = 1, l'arête courante appartient à un seul triangle (arête frontière) et les variables **a1, b1, c1, a2, b2, c2** et **de** sont des scalaires qui représentent les coordonnées et, pour **de**, la surface multipliée par deux du triangle numéro **numK** qui porte l'arête courante numéro **numA** de numéro local d'arête sur ce triangle **numALoc**.

Si **geomT.p_A** = 2, la variable **numK** est un vecteur colonne désignant les deux numéros de triangles auquel appartient l'arête courante numéro **numA**. L'ordre des triangles est celui donné par **Pf** (*cf.* 2.2.3) et le numéro local de l'arête numéro **numA**

sur le triangle numéro `numK(1)` (resp. `numK(2)`) est donné par `numAloc(1)` (resp. `numAloc(2)`).

Pour `geomT.p_A = 2`, les variables `a1`, `b1`, `c1`, `a2`, `b2`, `c2` et `de` sont alors des vecteurs colonnes représentant les coordonnées et, pour `de`, la surface multipliée par deux de chacun des deux triangle `numK(1)` et `numK(2)`.

La variable `uc` contient deux pages et `uc{:, :, 1}` (resp. `uc{:, :, 2}`) est la valeur aux points d'intégration, de coordonnées cartésiennes `Ac` et `Bc`, de l'arête de la restriction au premier (resp. second) triangle de la fonction et de ses dérivées d'ordre 1 représentée par `Uc`. Dans le cas d'un arête frontière, alors `numK(2)` est nul et `uc{:, :, 2}` correspond à la fonction nulle.

Les champs de `geomT` sont avec `m = geomT.p_A` qui vaut 1 ou 2 :

`Ac, Bc` : 1x1 double

`num_K` : 1x1 double

`numAlocT` : 1x1 double

`geomT` =

`nA` : [1x2 double] : sur l'arête, normale orientée par le premier triangle

`nT` : [mx2 double] : sur l'arête, normale extérieure pour chaque triangle,

`h_K` : [mx3 double] : tailles des cotés du triangle, pour chaque triangle,

`h_max` : [mx1 double] : diamètre de chaque triangle,

`h_A` : [1x1 double] : longueur de l'arête commune,

`p_A` : [1x1 double] : ici `p_A` vaut 1 pour une arête frontière et 2 sinon.

`num_K` : [mx1 double] : numéro global de triangle porté par l'arête

`som_A` : [mx2 double] : numéros des sommets de l'arête, triangle par triangle, l'ordre des numéros peut être inversé d'un triangle à l'autre,

`som_A_op` : [mx1 double] : Numéro du sommet opposé à l'arête, par triangle

`som_A_coord_X` : [mx2 double],

`som_A_coord_Y` : [mx2 double] : Coordonnées en x et y des sommets de l'arête pour chaque triangle,

`som_A_op_coord_X` : [mx1 double],

`som_A_op_coord_Y` : [mx1 double] : Coordonnées en x et y du sommet opposé à l'arête, pour chaque triangle.

La variable de sortie `Rc` est la variable de de type `cell` décrivant la valeur aux points de la formule de quadrature de la fonction à intégrer.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme : `varargin`. Si `varargin` est vide alors le terme `p` dans les appels ci-dessus pour le calcul en norme L^p , vaut `p = 2`.

- `varargin{1}`. Si `varargin` est vide alors $p = 2$, sinon $p = \text{varargin}\{1\}$:

`P DOUBLE ARRAY`. désigne p dans le calcul de la norme $L^p(\Omega)$.

- `varargin{2}`, `varargin{3}`. Termes `X` et `Y` de maillage vus dans les appels de fonction ci-dessus :

`X,Y : DOUBLE ARRAY`. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{5}`. Terme `T` de maillage vu dans les appels de fonction ci-dessus :

`T : DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

- `varargin{3}`. Terme `Tc` de *ddl* vu dans les appels de fonction ci-dessus :

`Tc : CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`S : DOUBLE ARRAY`. Valeur de la norme $L^p(\Gamma \cup \Sigma)$ de la fonction désignée par `nomf`.

Algorithme. Sur le principe de `norm_T` avec cette fois-ci une boucle sur la variable d'arêtes frontières `Pa` (*cf.* le Tableau 2.2.3-1b du Paragraphe 2.2.3) au lieu d'une boucle sur la variable de triangles `T` et sur le principe de la fonction `matrice_elem_Ad` au lieu de `matrice_elem_T`. Dans cet algorithme on prend en compte les valeurs de part et d'autres de l'arête courante qui portent sur la géométrie avec la variable `geomT` ci-dessus et les valeurs de la solution représentées par `uc`, avec ici pour `uc` un tableau de 2 pages.

Exemple. L'exemple suivant est tiré du `help` de la fonction `norm_Ad2` :

EXEMPLE 1

```
%Ci-dessous f1 est sur la frontiere la fonction erreur sur la
%solution multipliee par n.U avec n normale exterieure
%scalaire et U le champ qui transporte dans le probleme
%ci-dessous et vaut 0 ailleurs : on retrouve alors le meme
%resultat que dans l'exemple de NORM_AD. On a
%S1 = S2 = S3 = 3.9538e-004.
%Ensuite f2 est le saut de la derivee normale.
%On a S4 = 0.1774.
cd C :/EF_text/exemples/exemple_transport, instruct1_supg,
cd C :/EF_text/exemples/exemple_transport/equ/aut_norm
f1 = @eq_norm_Frb;%terme frontiere
```

```

f2 = @eq_norm_A2b;%saut des normales (geomT)
S1 = norm_Ad2(donnees, Uc, f1);%3.9538e-004
disp('erreur');
disp(S1);
load(donnees.nom_mail, 'X', 'Y', 'T', 'Front')
load(donnees.nom_prep, 'Tc')
S2 = norm_Ad2(donnees, Uc, f1, [], X, Y, T, Front);
disp('erreur');
disp(S2);
S3 = norm_Ad2(donnees, Uc, f1, [], X, Y, T, Front, Tc);
disp('erreur');
disp(S3);
S4 = norm_Ad2(donnees, Uc, f2);%0.1774
disp('erreur');
disp(S4);

```

Dans cet exemple, la première fonction de calcul de norme de `handle` de fonction `f1 = @eq_norm_Frb` est donnée par :

C :/EF_text/exemples/exemple-transport/equ/aut_norm/eq_norm_Frb (norm_Ad2)

```

function Rc = eq_norm_Frb(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA, geomT);
%Termes intermediaires :
Zero = zeros(1,length(Ac));
%Champ qui transporte :
f = donnees.champ_u;
[u1,u2] = feval(f,Ac,Bc);
%Valeur de n.u au bord, n normale unitaire exterieure :
if numF==1, nu = u2; end;
if numF==2, nu = u1; end;
if numF==3, nu = -u2; end;
if numF==4, nu = -u1; end;
if numF==0, nu = Zero; end;
%nu ci-dessous vaut |u.n| si u.n <= 0 et vaut 0 sinon :
nu = abs(nu).*(nu<=0);
Rc{1} = ((exp(-Ac) - uc{1,1,1})).*sqrt(nu));

```

Cette fonction renvoie des valeurs identiques à la fonction `eq_norm_Frb` utilisée à l'exemple précédent. La variable d'entrée supplémentaire `geomT` n'est pas utilisée ici.

Il est à noter que `uc` est ici une matrice de type cellules de 2 pages (*cf.* la présentation de la fonction dans ce Paragraphe, ainsi que le Paragraphe 2.5, sur l'utilisation de la fonction `norm_Ad` et le commentaire sur la fonction (`norm_Ad2`) qui suit).

Pour ce calcul d'erreur sur la frontière rentrante Γ_{β}^{-} on retrouve le même résultat que dans les exemples pour les fonctions `norm_A` et `norm_Ad`.

Ensuite, la seconde fonction de calcul de norme de `handle` de fonction `f2 = @eq_norm_A2b` est donnée par :

C :/EF_text/exemples/exemple_transport/equ/aut_norm/eq_norm_A2b (norm_Ad2)

```
function Rc = eq_norm_A2b(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA, geomT);
%Integration du saut sur les aretes interieures.
%Termes intermediaires :
Zero = zeros(1,length(Ac));
%Champ qui transporte :
f = donnees.champ_u;
[u1,u2] = feval(f,Ac,Bc);
Rc{1} = Zero;
%Valeur de n.u au bord, avec n normale unitaire exterieure
%  normale exterieure au premier triangle :
n = geomT.nA;
if numF==0
%Saut du gradient applique a la normale
Rc{1} = uc{1,2,1}*n(1) - uc{1,2,2}*n(1) + ...
        uc{1,3,1}*n(2) - uc{1,3,2}*n(2);
end;
```

Cette fonction permet de calculer la norme du saut de la fonction

$$\frac{\partial u}{\partial x}n_1 + \frac{\partial u}{\partial y}n_2$$

Dans la fonction `eq_norm_A2b`, `uc` contient deux pages qui sont `uc{:, :, 1}` et `uc{:, :, 2}`. Le terme `uc{:, :, 1}` donne la valeur aux points d'intégration de coordonnées cartésiennes `Ac` et `Bc` de l'arête courante de numéro `numA` de la restriction de u et de ses dérivées d'ordre 1 au triangle numéro `numK(1)`. Si cette arête est

portée par un second triangle alors le terme $uc\{:, :, 2\}$ donne cette fois-ci la valeur de la restriction de u au triangle numéro $numK(2)$. Dans le cas d'une arête frontière, alors $numK(2)$ est nul et $uc\{:, :, 2\}$ correspond à la fonction nulle (cf. Paragraphe 2.5, sur l'utilisation de la fonction `norm_Ad2`). Il est à noter qu'ici Ac et Bc concernent uniquement les points de la formule de quadrature sur l'arête et non, comme pour `norm_A` et `norm_Ad`, sur les 3 arêtes d'un triangle.

On pourra à la suite de l'exemple exécuter les commandes

```
cd C :/EF_text/exemples/exemple_transport/equ/aut_norm
f3 = @eq_norm_A2c;%saut suivant beta
S5 = norm_Ad2(donnees, Uc, f3);%0.1536
```

avec la fonction de calcul de norme de `handle` de fonction $f3 = @eq\_norm\_Adc$ donnée par :

C :/EF_text/exemples/exemple_transport/equ/aut_norm/eq_norm_A2c (norm_Ad2)

```
function Rc = eq_norm_A2b(donnees, numF, uc, a1, b1, c1, a2, ...
    b2, c2, de, Ac, Bc, numK, numALoc, numA, geomT);
%Integration du saut sur les aretes interieures.
```

```
%Termes intermediaires :
Zero = zeros(1,length(Ac));
%Champ qui transporte :
f = donnees.champ_u;
[u1,u2] = feval(f,Ac,Bc);
Rc{1} = Zero;
%Valeur de n.u au bord, avec n normale unitaire exterieure
%  normale exterieure au premier triangle :
n = geomT.nA;
if numF==0
%Saut suivant le champ beta = (1,0) :
beta = [1,0];
Rc{1} = (uc{1,2,1} - uc{1,2,2})*beta(1) + ...
    (uc{1,3,1} - uc{1,3,2})*beta(2);
end;
```

pour le calcul de la norme du saut de la dérivée de la solution suivant le champ β doonné par $\frac{\partial u}{\partial \beta}$.

Fonction :

```
function S = int_fonc_T(donnees, Uc, nomf, varargin) ;
```

Fonction utilisateur. Syntaxe analogue à `norm_T`, à l'exception de la première variable de `varargin`, `varargin{1}`, qui au lieu d'une valeur réelle est un `handle` ou un nom d'une fonction ϕ qui indique que l'on intègre la fonction $\phi(f)$.

La commande

```
S = int_fonc_T(donnees, Uc, nomf, nomf2) ;
```

calcule la valeur

$$\int_{\Omega} \phi(f)$$

où f est la fonction désignée par `nomf`, ϕ la fonction désignée par `nomf2`, avec `Uc` représentant un élément de l'espace EF produit E défini par `donnees.D.E` et `donnees.D.f`.

Les commandes suivantes n'utilisent pas les fichiers `donnees.nom_mail` et `donnees.nom_prep` :

```
S = int_fonc_T(donnees, Uc, nomf, nomf2, X, Y, T) ;
```

et, sans recalcul des degrés de liberté :

```
S = int_fonc_T(donnees, Uc, nomf, nomf2, X, Y, T, Tc) ;
```

Exemple. L'exemple suivant est tiré du `help` de la fonction `int_fonc_T` :

EXEMPLE 1

```
%On reprend l'exemple 1 de NORM_T et on retrouve le meme
%resultat avec le choix de PHI = f2 ci-dessous. Ci-dessous f1
%est la fonction erreur sur la solution exacte. On trouve
%S = sqrt(S2) = 2.8559e-004 et S2 = S^2 = 8.1562e-008.
cd C :/EF_text/exemples/exemple_transport, instruct1_supg,
cd C :/EF_text/exemples/exemple_transport/equ
f1 = @eq_norm_Dom ;
f2 = inline('abs(x).^ 2','x') ;
S = norm_T(donnees, Uc, f1) ;
%Calcul identique a la racine du total pres
S2 = int_fonc_T(donnees, Uc, f1, f2) ;
disp(S - sqrt(S2)) ;
```

Fonction :

```
function [Sc, Fc] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;
```

Transformations d'une matrice et d'un vecteur par blocs représentés par des tableaux de cellules *Sc* et *Fc*. Dans la fonction *EF_assemblage* du code cette transformation correspond à la prise en compte des conditions au bord.

La commande

```
[Sc,Fc] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;
```

transforme *Sc* et *Fc* de la manière suivante.

Soient *S* la matrice et *F* le vecteur de type *DOUBLE ARRAY* ou *SPARSE ARRAY* obtenus à partir de leur représentation par blocs *Sc* et *Fc* qui sont de type *CELL ARRAY*.

Soit *Nbv* le vecteur obtenu par concaténation des cellules du tableau de type *CELL* *Nb* comme décrit ci-dessous (*cf.* aussi le code de la fonction) :

```
Nbv = [Nb{1}( : ) ; Dnc(1) + Nb{2}( : ) ;  
      ... Dnc(1) + Dnc(2) + Nb{3}( : ) ;  
      ... sum(Dnc(1 :end-1)) + Nb{end}( : ) ],
```

avec *Dnc* vecteur ligne.

Dans le code *EF*, on passe ainsi d'une numérotation par espace d'*EF* des *ddl* sur lesquels portent les conditions au bord à une numérotation globale.

Soit *Sbv* le vecteur de même taille que *F* obtenu par concaténation du tableau de cellules *Sb* comme décrit ci-dessous (*cf.* aussi le code de la fonction) :

```
Sbv = [Sb{1} ; Sb{2} ; Sb{3}, ... ; Sb{end}].
```

Alors *S* et *F* subissent la transformation suivante :

```
L = -S*Sbv ;  
F = L + F ;  
F(Nbv) = Sbv(Nbv) ;
```

et mise à 0 (ou 1 pour le terme diagonal) des lignes et colonnes indiquées par *Nbv* :

```
S = transf_mat1(S,Nbv) ;
```

Dans le code *EF* ces transformations sont décrites à la fin du Paragraphe 3.2.1 où sont données les formules donnant *S* et *F* (*cf.* les variables *S* et *F* dans ce Paragraphe). Le lien avec ces formules et la fonction *systeme_Bord_cell* est donné au Paragraphe 3.2.3.

Ensuite *S* et *F* sont reconvertis en tableau de cellules au même format que *Sc* et *Fc*.

Dans la fonction *EF_assemblage*, la fonction *systeme_Bord_cell* transforme le système $S \cdot X = F$ pour tenir compte des conditions au bord pour chaque variable, *Sc* et *Fc* étant la représentation par blocs de *S* et *F*.

4.3. DESCRIPTION DES FONCTIONS FONCTION `Assemblage/systeme_Bord_cell`

$\text{Nb}\{i\}$ est un ensemble de numéros de *ddl* de E_i le i -ème espace d'EF de l'espace EF produit E sur lequel on résout. Ces *ddl* sont placés sur la frontière du domaine et on a une condition aux limites en ces *ddl*.

$\text{Sb}\{i\}(j)$ est la valeur imposée sur la i -ème composante sur E de la solution du problème EF au *ddl* numéro j . $\text{Sb}\{1\}(j)$ est la valeur de cette composante au *ddl* numéro i et vaut 0 sinon ou un nombre quelconque.

Remarque 1. $\text{Nb}\{i\}$ est de dimension inférieure à la dimension de E_i alors que $\text{Sb}\{i\}$ est égal à la dimension de E_i . Seule la valeur de $\text{Sb}\{i\}(\text{Nb}\{i\})$ est prise en compte dans cette fonction (valeur de la fonction aux *ddl* sur lesquels on a une condition imposée).

Remarque 2. Dans l'algorithme de `systeme_Bord_cell` la valeur de la variable `D` n'est pas utilisée, `D` peut être mise égale à []. On pourrait aussi utiliser les dimensions de `Fc` à la place de `Dnc` et ne pas utiliser la variable `Dnc` comme variable d'entrée.

Variables d'entrée :

`D` : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1. Non utilisée. Peut être mise à [] dans cette fonction.

`Sc` : `CELL ARRAY`. Représentation par bloc d'une matrice carrée, sous forme de tableau de cellules. Correspond dans la fonction `EF_assemblage` du code à la matrice du système.

`Fc` : `CELL ARRAY`. Vecteur colonne. Représentation par bloc d'un vecteur colonne, sous forme de tableau de cellules. Correspond dans la fonction `EF_assemblage` du code au second membre du système.

`Dnc` : `DOUBLE ARRAY`. Vecteur ligne. Délimite la taille des blocs. $\text{Dnc}(i)$ est le nombre de degrés de liberté du i -ème espace d'EF E_i de l'espace EF produit E .

`Nb` : `CELL ARRAY`. Vecteur colonne. $\text{Nb}\{i\}$ est un vecteur colonne de type `DOUBLE ARRAY` ou vaut []. Dans la fonction `EF_assemblage` ce vecteur correspond aux numéros de *ddl* du i -ème espace de l'espace EF produit E , sur lesquels on a une condition de type imposée sur la i -ème variable, le même numéro de *ddl* peut figurer plusieurs fois dans $\text{Nb}\{i\}$. Dans l'appel par la fonction `EF_assemblage`, cette variable est la première variable de sortie de la fonction `cb`.

`Sb` : `CELL ARRAY`. Vecteur colonne. $\text{Sb}\{i\}$ est un vecteur colonne de taille $\text{Dnc}(i)$. Dans la fonction `EF_assemblage` ce vecteur correspond aux *ddl* pour lesquels on a une condition imposée pour la i -ème variable en donnant la valeur de la condition en ces *ddl* : la valeur imposée sur la solution au *ddl* numéro $\text{Nb}\{i\}(j)$ est donnée par $\text{Sb}\{i\}(\text{Nb}\{i\}(j))$. Dans l'appel par la fonction `EF_assemblage`, cette variable est la seconde variable de sortie de la fonction `cb`.

Variables de sortie :

Sc : CELL ARRAY. Représentation par blocs d'une matrice carrée, sous forme de tableau de cellules. Correspond dans la fonction `EF_assemblage` du code à la nouvelle matrice du système qui prend en compte les conditions au bord.

Fc : CELL ARRAY. Représentation par blocs d'un vecteur, sous forme de tableau de cellules. Correspond dans la fonction `EF_assemblage` du code au nouveau second membre du système qui prend en compte les conditions au bord.

Exemple 1 : Cet exemple est tiré du `help` de la fonction. Dans cet exemple les variables d'entrée de `systeme_Bord` sont pour **Dnc** :

$$\mathbf{Dnc} = (3, 3),$$

ensuite, la matrice **Sc** est la matrice 2×2 par blocs 3×3 donnée par

$$\mathbf{Sc}\{1, 1\} = \mathbf{Sc}\{1, 2\} = \mathbf{Sc}\{2, 1\} = \mathbf{Sc}\{2, 2\} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 3 & 6 \end{pmatrix},$$

et le vecteur **Fc** est le vecteur 2×1 par blocs 3×1 donnée par

$$\mathbf{Fc}\{1\} = (2, 3, 4), \mathbf{Fc}\{2\} = (7, 8, 9).$$

Ensuite, les valeurs de **Nb** et **Sb** choisies sont :

$$\mathbf{Nb}\{1\} = (1, 2), \mathbf{Nb}\{2\} = 1, \mathbf{Sb}\{1\} = (10, 20, 0), \mathbf{Sb}\{2\} = (30, 0, 0).$$

Dans l'algorithme la matrice **S** et le vecteur **F** ci-dessus prennent alors la valeur :

$$\mathbf{S} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 3 & 6 & 1 & 3 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 3 & 6 & 1 & 3 & 6 \end{pmatrix}, \mathbf{F} = \begin{pmatrix} 2 \\ 3 \\ 4 \\ 7 \\ 8 \\ 9 \end{pmatrix},$$

et les variables **Nbv** et **Sbv** la valeur

$$\mathbf{Nbv} = (1, 2, 4), \mathbf{Sbv} = (10, 20, 0, 30, 0, 0)'.$$

Ci-dessus $\mathbf{Nbv}([1, 2]) = \mathbf{Nb}\{1\}$ et $\mathbf{Nbv}(2) = \mathbf{Dnc}(1) + \mathbf{Nb}\{2\} = 4$. En sortie de la fonction `system_Bord_cell` on obtient donc la matrice notée **Sc2** obtenue en mettant à 0 les lignes et colonnes $(1, 2, 4)$ de **S**, excepté les termes diagonaux qui

4.3. DESCRIPTION DES FONCTIONS FONCTION `Assemblage/systeme_Bord_cell`

sont posés à 1, puis en mettant la matrice obtenue sous forme de matrices 2×2 par blocs 3×3 :

$$\begin{aligned} \text{Sc2}\{1,1\} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 6 \end{pmatrix}, \quad \text{Sc2}\{1,2\} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 3 & 6 \end{pmatrix}, \\ \text{Sc2}\{2,1\} &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 0 & 6 \end{pmatrix}, \quad \text{Sc2}\{2,2\} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 3 & 6 \end{pmatrix}. \end{aligned}$$

Ensuite on obtient le vecteur notée `Fc2` obtenu en calculant, comme ci-dessus dans la description de la fonction, le terme

$$-S * \text{Sbv} + F = (-58, -77, -96, -53, -72, -91)' = G,$$

puis en posant à 10, 20, 30 les termes 1, 2 et 4 du vecteur G obtenu, ce qui donne le vecteur G_2 :

$$G_2 = (10, 20, -96, 30, -72, -91)'$$

et finalement en mettant G_2 sous forme de vecteurs 2×1 par blocs 3×1 :

$$\text{Fc2}\{1\} = (10, 20, -96)'; \quad \text{Fc2}\{2\} = (30, -72, -91)'.$$

L'exemple réalise d'abord avec `system_Bord_cell` le calcul de `Sc2` et `Fc2` et affiche les matrices correspondantes, puis effectue aussi, à titre d'exemple, le calcul de S , F , Nbv et de $S * \text{Sbv} + F$.

Les commandes de cet exemple tirées du `help` de la fonction `systeme_Bord_cell` sont les suivantes :

EXEMPLE 1

```
%Cet exemple met en oeuvre la transformation effectuee
%par systeme_Bord_cell sur une matrice et un vecteur donne.
clear

%Initialisation de D, Dnc, Sc, Fc, qui correspond a deux
%espaces de dimension 3
D = [] ;
Dnc = [3,3] ;
Sc = {pascal(3), pascal(3) ; pascal(3), pascal(3)} ;
Fc = {[2 :4]' ; [7 :9]}' ;

%Initialisation de Nb : on aura Nbv = [1,2,0,1,0,0]
```

```

Nb = {[1,2] ; [1]} ;

%Initialisation de Sb : on aura Sbv = [10,20,0,30,0,0]
Sb = {[10,20,0]' ; [30,0,0]}' ;

%Utilisation de la fonction
[Sc2,Fc2] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;

%Mise de la matrice sous forme de matrice de cellules :
S2 = trans_mat_cell_mat(Sc2) ;
F2 = trans_mat_cell_mat(Fc2) ;

%Affichage des resultats ;
disp('Nouvelle matrice et sd membre sous forme matricielle :')
disp(['Sur S2 on a des 0 et des 1 pour les termes Nb{1}' ...
      ' et Dnc(1) + Nb{2} ie les termes (1,2,4) :']) ;
disp('S2') ;
disp(S2) ;
disp(['F2 vaut Sb{1}(Nb{1}) et Sb{2}(Nb{2}) aux ', ...
      ' points Nb{1} et 3 + Nb{2} ie les termes (1,2,4) et', ...
      ' -S*Sbv + F aux autres termes']) ;
disp('F2') ;
disp(F2) ;
disp('Detail de calcul :on calcule S, F, Sbv puis -S*Sbv + F') ;
S = trans_mat_cell_mat(Sc) ;
F = trans_mat_cell_mat(Fc) ;
Sbv = [Sb{1} ; Sb{2}] ;
-S*Sbv + F ;
disp('-S*Sbv + F') ;
disp(-S*Sbv + F) ;

```

Exemple 2.

Cet exemple est donné par la commande (voir aussi cet exemple dans le help de la fonction) :

```
cd C :/EF/exemples/Assemblage, exemple_systeme_Bord_cell
```

Il traite le cas d'un système (découplé) sur le domaine $\Omega =]0, 1[^2$ de frontière Γ :

$$-\Delta u_1 = 2x(1-x) + 2y(1-y), \quad u_1 = 0 \text{ sur } \Gamma,$$

$$-\Delta u_2 = 2x(1-x) + 2y(1-y), \quad u_2 = 0 \text{ sur } \Gamma,$$

de solution

$$u_1(x, y) = u_2(x, y) = x(1-x)y(1-y).$$

Le domaine Ω est muni du maillage du carré unité par 8 triangles de la figure 4.3-5 et on pose comme espace EF produit E pour l'approximation du problème : $E = E_1 \times E_2$, où l'on a pris E_1 et E_2 égaux à $F_1 = P_4^c$, le P_4 continu. Dans cet exemple on procède à titre illustratif, sans utiliser `EF_assemblage`, de la manière suivante. On calcule avec la fonction `EF_Prep` le fichier `.mat` relatif aux espaces d'EF et aux formules de quadrature (*cf.* la description de ce type de fichier au Paragraphe 2.2.3) puis on assemble le système directement à partir de ce fichier. Le fichier obtenu contient la variable `SL_Dom` du Tableau 2.2.3-1a du Paragraphe 2.2.3. Les cellules de `SL_Dom` contiennent des matrices déjà assemblées qui suffisent ici pour calculer la matrice et le second membre du système, sans passer par la fonction d'assemblage `EF_assemblage`.

Ainsi, soit $\{\Phi_i\}$, la base de F_1 , les valeurs prises dans l'exemple ci-dessous par les matrices

$$\text{SL\_Dom}\{1, 1, 1, 1\}, \text{SL\_Dom}\{1, 1, 2, 2\} \text{ et } \text{SL\_Dom}\{1, 1, 3, 3\}$$

sont respectivement les matrices dont les terme (i, j) sont donnés respectivement par (*cf.* Tableau 2.2.3-1a) :

$$\int_{\Omega} \Phi_j \Phi_i, \int_{\Omega} \Phi_{j,x} \Phi_{i,x}, \int_{\Omega} \Phi_{j,y} \Phi_{i,y}.$$

Ensuite, comme le second membre du système est dans E et que la base de F_1 est nodale, on vérifie que le second membre du système est donné par :

$$(\text{SL\_Dom}\{1, 1, 1, 1\}F, \text{SL\_Dom}\{1, 1, 1, 1\}F)$$

où F vérifie

$$F_i = x_i(1-x_i)y_i(1-y_i)$$

où $\{(x_i, y_i)\}$ désigne la famille des *ddl* du P_4^c .

Le *script* de cet exemple est le suivant.

`EF/exemples/Assemblage/exemple_systeme_Bord_cell.m`

`%Fichier EF/exemples/Assemblage/exemple_systeme_Bord_cell.m`

`%Utilisation de systeme_Bord_cell : calcul d'un systeme EDP sur`

```

% l'espace  $E = E_1 \times E_2 = P_4 \times P_4$  continu. On resout sur le carre
% unite :
%
%      - Delta  $u_1 = 2y(1 - y) + 2x(1 - x)$ ,  $u_1 = 0$  au bord
%      - Delta  $u_2 = 2y(1 - y) + 2x(1 - x)$ ,  $u_2 = 0$  au bord
%
% de solution  $u_1(x,y) = u_2(x,y) = x(1 - x)y(1 - y)$ .
% Ici, les matrices calculees par EF_Prep, suffisent pour
% assembler le systeme. Pour illustrer la fonction
% systeme_Bord_cell, on peut aussi passer par EF_assemblage
% qui appelle cette fonction.
clear ;

%Choix des types d'espaces :  $F_1 = P_4$  continu :
D.E = [4] ;

%Choix de l'espace produit  $E = E_1 \times E_2 = F_1 \times F_1 = P_4 \times P_4$ 
D.f = [1,1] ;

%Construction du maillage : triangles et coordonnees T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;

%et la frontiere Front :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Choix des formules d'integration :
D.intT = 10 ;
D.intA = 6 ;

```

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage/systeme_Bord_cell

```
%Derivations (fixe dans le code) :
D.der = [0,1,2] ;

%Initialisation des champs de la variable donnees necessaires a
%   EF_Prep :
donnees.D = D ;

donnees.nom_mail = ...
'C :/EF/exemples/Assemblage/fich_exemple_systeme_Bord_cell/f_mail' ;
donnees.nom_prep = ...
'C :/EF/exemples/Assemblage/fich_exemple_systeme_Bord_cell/f_prep' ;

%Creation du fichier de maillage :
save(donnees.nom_mail,'T','X','Y','Front','Noeu')

%Calcul du fichier relatif au espaces d'EF et aux formules
%   d'integration :
EF_Prep(donnees) ;

%Chargement de la matrice SL_Dom et des coordonnees des DDL :
load(donnees.nom_prep,'Xc','Yc','SL_Dom') ;

%Nombres de degres de liberte de F1 :
n_ddl = length(Xc{1}) ;

%Matrice du systeme. On verifie qu'elle est donnee par :
Sc{1,1} = SL_Dom{1,1,2,2} + SL_Dom{1,1,3,3} ;
Sc{1,2} = zeros( size(SL_Dom{1,1,2,2}) ) ;
Sc{2,1} = zeros( size(SL_Dom{1,1,2,2}) ) ;
Sc{2,2} = SL_Dom{1,1,2,2} + SL_Dom{1,1,3,3} ;
%
%Ci-dessus les deux premiers indices font reference au type
%   d'espace EF choisi : ici il y en a un, F1 (P4 continu), et les
```

```

% deux indices suivants aux derivations : 1, pas de derivation,
% 2 : derivation par rapport a x, 3 : derivation par rapport a y.

%Second membre. Celui-ci etant P4, on verifie qu'il est alors donne
% par :
SF      = SL_Dom{1,1,1,1} ;
Fc{1}   = SF*[2*Xc{1}.*(1 - Xc{1}) + 2*Yc{1}.*(1 - Yc{1})]' ;
Fc{2,1} = SF*[2*Xc{1}.*(1 - Xc{1}) + 2*Yc{1}.*(1 - Yc{1})]' ;

%Variable Dnc des dimensions des espaces successifs de l'espace
% produit :
Dnc = [n_ddl,n_ddl] ;

%Tableau des DDL situes sur la frontiere : la methode simplifiee
% utilisee ci-dessous fonctionne s'il n'y a pas d'erreur :
% d'arrondi, ce qui est le cas ici
I = find(Xc{1}==0 | Xc{1}==1 | Yc{1}==0 | Yc{1}==1) ;

%Calcul des parametres d'entree Nb et Sb de systeme_Bord_cell :
%1) DDL du premier espace EF E1 ou l'on a une condition aux
% limites :
Nb{1} = I' ;

%2) DDL du second espace EF E2 ou l'on a une condition aux
% limites :
Nb{2,1} = I' ;

%Valeur imposee de u1 sur les DDL frontieres du premier espace E1
% pour lesquels figure une condition aux limites.
% Sb{1}(I) est la valeur de u1 aux DDL de numero I concernees
% (valeur 0 ici) et vaut 0 aux autres DDL :
Sb{1} = zeros(n_ddl,1) ;

%Valeur aux DDL du second espace E2 (condition nulle au bord) :

```


4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage/systeme_Bord_cell

```
Sb{2,1} = zeros(n_ddl,1) ;

%Utilisation de systeme_Bord_cell :
[Sc2,Fc2] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;

%Construction et resolution du systeme :
S = [Sc2{1,1}, Sc2{1,2} ; Sc2{2,1}, Sc2{2,2}] ;
F = [Fc2{1} ; Fc2{2}] ;
U = S\F ;

%Mise de la solution sous forme de tableau de cellules :
Uc{1} = U(1 :n_ddl) ;
Uc{2,1} = U(n_ddl + 1 :end) ;

%Comparaison aux DDL de la solution approchee et de la solution
%exacte :
disp(['max(abs(erreur)) pour chaque composante de la' ...
      ' solution, le max est pris aux ddl (9.7145e-017) :']) ;
disp(max(Uc{1}' - Xc{1}.*(1 - Xc{1}).*Yc{1}.*(1 - Yc{1}))) ;
disp(max(Uc{2}' - Xc{1}.*(1 - Xc{1}).*Yc{1}.*(1 - Yc{1}))) ;
```

Algorithme :

EF/Assemblage/systeme_bord_cell.m

On remet ici la partie déclarative de la fonction :

```
function [Sc, Fc] = systeme_Bord_cell(D, Sc, Fc, Dnc, Nb, Sb) ;
```

Note de programmation. Cette fonction est utilisée dans EF_assemblage avant élimination d'inconnues par cellules.

- Tests sur les variables d'entrée.
- Mise des matrices de cellules sous forme de matrice :

```
S = trans_mat_cell_mat(Sc) ;
```

```
F = trans_mat_cell_mat(Fc) ;
```

```
Sbv = trans_mat_cell_mat(Sb) ;
```

- Calcul de Nbv :

```
Nbv = [] ;
```

```
Pour i=1 :length(Sb)
    Si ~isempty(Nb{i})
        Nbv = [Nbv ; sum(Dnc(1 :i)) - Dnc(i) + Nb{i}( :)] ;
    Fin ;
Fin ;
```

- Transformation du vecteur :

```
L = -S*Sbv ;
F = L + F ;
F(Nbv) = Sbv(Nbv) ; %Nbv peut etre vide
```

- Transformation de la matrice avec placement de 0 et de 1 :

```
S = transf_mat1(S,Nbv) ;
```

- Transformation des matrices en matrices de cellules :

```
Sc = trans_mat_mat_cell(S,D,Dnc) ;
Fc = trans_mat_mat_cell(F,D,Dnc) ;
```

Fonction :

```
function S2 = transf_mat1(S, n) ;
```

La matrice **S2** est la matrice obtenue en plaçant, pour $1 \leq i \leq \text{length}(n)$, des 0 sur la ligne et la colonne $n(i)$ de la matrice carrée **S** excepté pour le terme diagonal qui lui est mis à 1 et en mettant la matrice obtenue dans **S2**. Si **n** est vide **S2** est alors égale à **S**.

La variable **n**, peut avoir des composantes $n(i)$ plus grande que la taille de **S**, cela ne modifie alors pas **S2** en sortie.

On pourra aussi modifier la fonction **transf_mat1**, pour économie de mémoire, en posant :

```
function S = transf_mat1(S, n) ;
```

et supprimer la dernière instruction

```
S2 = S ;
```

La variable **S**, qui est en entrée, ne sera pas modifiée par un appel du type

```
S2 = transf_mat1(S, n) ;
```

Variables d'entrée :

N : DOUBLE ARRAY. Vecteur ligne ou colonne d'entiers.

S : DOUBLE ARRAY. Matrice carrée à modifier

Variables de sortie :

S2 : DOUBLE ARRAY. Matrice carrée modifiée.

Exemple : Cet exemple est tiré du **help** de la fonction **transf_mat1**.

EXEMPLE

```
S = pascal(6) ;  
S1 = transf_mat1(S,[1,2,4]) ;  
S2 = transf_mat1(S,[1,3]) ;  
S3 = transf_mat1(S,[5,3,1]) ;  
S4 = transf_mat1(S,[5,3,1]') ;
```

Algorithme :

$S2$ est calculé par la formule

$$S2 = F * S * F + E$$

où E et F sont des matrices de la taille de $S = \mathbf{S}$ et telles que E est la matrice nulle excepté pour les termes diagonaux de numéro $\mathbf{n}(i)$ inférieur ou égal à la taille de S , qui valent 1 et où F est la matrice identité excepté pour les termes diagonaux de numéro $\mathbf{n}(i)$ inférieur ou égal à la taille de S , qui valent 0.

Par exemple le choix

$$S = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 3 & 6 & 10 & 15 & 21 \\ 1 & 4 & 10 & 20 & 35 & 56 \\ 1 & 5 & 15 & 35 & 70 & 126 \\ 1 & 6 & 21 & 56 & 126 & 252 \end{pmatrix} \text{ et } n = (1, 2, 4);$$

donne

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad F = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

et

$$S2 = \text{transf\_mat1}(S, n) = F * S * F + E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 15 & 21 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 15 & 0 & 70 & 126 \\ 0 & 0 & 21 & 0 & 126 & 252 \end{pmatrix}.$$

Ce calcul est réalisé par la commande

```
S1 = transf_mat1(S, [1,2,4]) ;
```

de l'exemple du `help` de la fonction ci-dessus.

Fonctions de `trace_graphe_interp`.**Fonction :**

```
function [X2, Y2, T2, U, varargout] = trac_solEc(donnees, Uc, ...
                                                der, r, M, varargin);
```

Cette fonction permet de tracer les fonctions de l'espace EF produit E , en particulier les solutions des problèmes EF, elle permet aussi de tracer des champs de vecteurs. A l'exception de l'utilisation de la fonction `pdesurf` de `pdetool`, cette fonction est identique à la fonction `trac_solEd`, seul l'aspect des surfaces diffère.

L'appel de fonctions du code peut se faire de manière transparente ou non dans le sens suivant. Par exemple, pour tracer une solution EF, l'appel :

```
[X2, Y2, T2, U2] = trac_solEc(donnees, Uc, der, r, M);
```

utilise les variables de maillage `X`, `Y`, `T` et la variable relative aux espaces d'éléments finis `Tc` stockées dans les fichiers `.mat` désignés par les champs `donnees.nom_prep` et `donnees.nom_mail`, alors que les deux appels «transparents» suivants :

```
[X2, Y2, T2, U2] = trac_solEc(donnees, Uc, ...
                              der, r, M, [], [], X ,Y, T, Tc);
[X2, Y2, T2, U2] = trac_solEc(donnees, Uc, ...
                              der, r, M, [], [], X ,Y, T);
```

n'utilisent aucun fichier `.mat`.

La commande

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, der, r, M);
```

trace pour la variable de dérivation `der` = [] ou 0, les composantes `r(1)`, `r(2)`, *etc.* de la fonction de l'espace EF produit E décrit par la variable `donnees` et représentée par la variable `Uc`. Le tracé se fait en calculant les valeurs de la fonction aux sommets d'un maillage obtenu par raffinement du maillage utilisé pour l'approximation EF, `M` étant la variable de raffinement.

Les champs de la variable `donnees` à initialiser sont `donnees.nom_prep` et `donnees.nom_mail` qui désignent respectivement le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) ainsi que le champ `donnees.D.E` et `donnees.D.f` de la variable de description des espaces d'EF `donnees.D` décrite au Tableau 2.2.1-1c, Paragraphe 2.2.1.

Pour `der` = 1 ou 2, c'est la dérivée par rapport à x ou y des fonctions qui est tracée. Cette dérivation est en fait la dérivée de la restriction des fonctions à chaque triangle du maillage, les fonctions n'étant en général pas différentiables aux points situés sur les arêtes.

Le raffinement s'obtient par subdivision de chaque triangle du maillage en M^2 sous-triangles égaux. La numérotation des sommets est la suivante. Les sommets des

sous-triangles du triangle numéro 1 sont numérotés de 1 à $(M+2)(M+1)/2$, ceux du triangle numéro 2 de $1 + (M+2)(M+1)/2$ à $(M+2)(M+1)$ *etc.* On remarque qu'il ne s'agit donc pas d'une structure de maillage de type maillage EF utilisée dans le code. Un exemple de triangle raffiné avec $M = 4$ est donné à la figure 4.2-1 du Paragraphe 4.2.

Les variables de sortie **X2** et **Y2** sont les vecteurs lignes des coordonnées des sommets du maillage raffiné et **T2** le tableau des numéros de sommets de ce maillage, triangle par triangle. Si i est une valeur de **r**, $U2\{i\}(j)$ est la valeur aux sommets de coordonnées $(X2(i), Y2(i))$ du maillage raffiné de la fonction (ou de la dérivée spécifiée de cette fonction) de composantes $Uc\{i\}$ dans la base de E_i , le i -ème espace de l'espace E . La commande

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, r, M, nfig);
```

permet de spécifier avec la variable optionnelle **nfig** le numéro de la première des figures, ce numéro vaut 1 par défaut ou lorsque **nfig** = []. Il est aussi possible de tracer le graphe des fonctions dans une seule figure contenant un tableau de $m \times n$ graphes en spécifiant les champs **nfig.n**, le numéro de figure et **nfig.t** = $[m, n]$ le vecteur (ligne ou colonne) précisant les dimensions du tableau. Pour effectuer les opérations de calcul de **X2**, **Y2**, **T2** et **U** sans tracer les fonctions on posera **nfig** = 'n'.

La commande

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, r, M, nfig, vec);
```

où **vec** est soit vide, soit une matrice de 2 colonnes, permet de tracer pour chaque valeur de k les champs de vecteurs formés des composantes $vec(k, 1)$ et $vec(k, 2)$ de la fonction représentée par **Uc**. Les valeurs de **U** correspondantes à ces composantes sont aussi calculées.

La commande

```
[X2, Y2, T2, U] = ...
    trac_solEc(donnees, Uc, 0, r, M, nfig, vec, X, Y, T);
```

où **X**, **Y** et **T** sont les variables du maillage (*cf.* Tableau 2.2.2-1, Paragraphe 2.2.2) permet d'effectuer les mêmes opérations que ci-dessus mais sans utiliser les fichiers désignés par **donnees_nom_prep** et **donnees_nom_mail** qui sont alors facultatifs.

La commande

```
[X2, Y2, T2, U] = ...
    trac_solEc(donnees, Uc, 0, r, M, nfig, vec, X, Y, T, Tc);
```

où **Tc** est le tableau des *ddl* du Tableau 2.2.3-1b du Paragraphe 2.2.3 effectue la même opération que la précédente. Elle est plus rapide car elle ne nécessite pas le calcul de la variable de **Tc** qu'effectue la commande précédente.

Enfin les commandes du type

```
[X2, Y2, T2, U, Tc] = trac_solEc(donnees, Uc, r, M, ...);
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

ou

```
[X2, Y2, T2, U, Tc, X, Y, T] = ...  
trac_solEc(donnees, Uc, r, M, ...);
```

permettent de récupérer la variable optionnelle de sortie `Tc`, tableau des numéros de `ddl` ainsi que, pour la deuxième commande, les variables de maillage `X`, `Y` et `T`.

Variables d'entrée :

`donnees` : `STRUCT ARRAY`. Variable décrivant les données du problème EF. Pour les appels du type :

```
trac_solEc(donnees, Uc, der, r, M);  
trac_solEc(donnees, Uc, der, r, M, nfig);  
trac_solEc(donnees, Uc, der, r, M, nfig, vec);
```

les champs à initialiser sont

```
donnees.D.E,  
donnees.D.f,  
donnees.nom_prep,  
donnees.nom_mail.
```

Et pour les appels du type :

```
trac_solEc(donnees, Uc, der, r, M, nfig, vec, X, Y, T, ...);
```

avec appel des variables de maillage `X`, `Y`, `T` et éventuellement de `Tc`, les champs à initialiser sont :

```
donnees.D.E,  
donnees.D.f.
```

Dans ce cas la fonction `trac_solEc` n'utilise pas le fichier relatif aux espaces d'EF et aux formules d'intégration et le fichier de maillage (*cf.* Tableau 2.2.1-1a, Paragraphe 2.2.1) qui contiennent des variables de maillage `X`, `Y`, `T` et de `ddl` dont la variable `Tc`.

`Uc` : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux `ddl` correspondants sont identiques. Les cellules i de `Uc` peuvent être vides si i n'est pas dans `r` ou `vec`.

`der` : `DOUBLE ARRAY`. Les valeurs de `der` = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

r : DOUBLE ARRAY. Variable portant sur les composantes de l'espace EF produit E . Vecteur ligne, $r(i)$ désigne la composante de la fonction à tracer. Si $r = []$ et si **vec** n'est pas utilisée ou est vide alors les cellules de **Uc** peuvent être toutes vides. La variable **r** doit être un vecteur ligne quand elle est utilisée avec **vec**.

M : DOUBLE ARRAY. Variable de raffinement de maillage pour le tracé de fonctions de l'espace EF produit E . Chaque triangle du maillage est subdivisé en M^2 sous-triangles égaux. L'ordre de numérotation des triangles et des sommets est donné pour le cas $M = 4$ sur la figure 4.2-1 du Paragraphe 4.2.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme :

Première variable optionnelle :

nfig : DOUBLE ARRAY ou STRUCT ARRAY. Variable de figure. Lorsque **nfig** est de type DOUBLE ARRAY, la composante $r(i)$ de la fonction à tracer est tracée sur la figure numéro i si **nfig** = [] et sur la figure numéro **nfig**-1+ i si **nfig** est un entier. Lorsque **nfig** est de type STRUCT ARRAY, **nfig.n** indique le numéro de figure sur lesquelles seront tracés tous les graphes dans un tableau de **nfig.t**(1) $\times$ **nfig.t**(2) figures.

Seconde variable optionnelle :

vec : DOUBLE ARRAY. Variable pour les tracés de champs de vecteur. Matrice de deux colonnes. Pour le tracé du champs de vecteurs donnés par les composantes **vec**(k , 1) et **vec**(k , 2) de la fonction représentée par **Uc**. Peut prendre la valeur [].

Troisième et quatrième variable optionnelle :

X,Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Cinquième variable optionnelle :

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Sixième variable optionnelle :

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice **Tc**{ i } décrit triangle par triangle les ddl du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

X2,Y2 : DOUBLE ARRAY. Coordonnées des sommets du maillage obtenu par raffinement du maillage de l'espace EF produit courant E . Le maillage raffiné n'est pas de type EF. Le maillage de chaque triangle en sous-triangles est de type maillage EF.

T2 : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage raffiné. Les numéros de sommets du triangle k vont de $1 + (k - 1)(M + 2)(M + 1)/2$ à $k(M + 2)(M + 1)/2$, cette numérotation n'est pas de type maillage EF.

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

U : CELL ARRAY. Vecteur colonne de même taille que **U**. Le terme $U\{i\}$ est soit vide, soit, lorsqu'il a été calculé, le vecteur colonne telle que $U\{i\}(j)$ est la valeur au point $(X2(i), Y2(i))$ de la i -ème composante de la fonction (ou de l'une de ses dérivées, si $der = 1$ ou 2) représentée par **Uc**.

Variables de sortie optionnelles :

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice $Tc\{i\}$ décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

X,Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Exemples :

On donne deux exemples donnés par les fichiers *scripts exemple_trac_solEc.m* et *exemple2_trac_solEc.m* du répertoire `EF/exemples/trace_graphe_interp`.

Dans ces deux exemples le domaine Ω est muni du maillage du carré unité par 8 triangles de la figure 4.3-5. Le premier exemple utilise le fichier de maillage et le fichier relatifs aux EF et aux formules de quadrature. Le second utilise directement les variables de maillage et de *ddl* et utilise aussi l'affichage par tableaux de figures.

Dans ces deux exemples, les espaces d'EF sont identiques excepté l'utilisation d'un espace d'EF avec fonctions de type bulles dans le deuxième exemple. On notera alors dans cet exemple l'utilisation de la fonction `inv_interp_EF_ddl` pour le calcul des composantes d'une fonction d'un espace de type bulles à partir de ses valeurs aux *ddl* (on est dans le cas d'une base qui n'est pas nodale).

Exemple 1.

La commande pour cet exemple est

```
cd C :/EF/exemples/trace_graphe_interp, exemple_trac_solEc ;
```

Dans cet exemple on pose comme espace EF produit E :

$$E = P_1^c \times P_2^c \times P_2^c \times P_5^c \times P_3^d,$$

et on trace les composantes de la fonction $f = (f_1, f_2, f_3, f_4, f_5) \in E$ donnée par :

$$\begin{aligned} f_1(x, y) &= x + y, \quad f_2(x, y) = -y, \quad f_3(x, y) = x, \quad f_4(x, y) = I_5(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4), \end{aligned}$$

où pour une fonction f continue $I_5(f)$ est la fonction du P_5^c qui interpole f aux *ddl* du P_5^c et $I_3(f)$ est la fonction du P_3^d qui interpole f aux *ddl* du P_3^d . Dans

le fichier exemple, les composantes des fonctions f_i , utilisées pour effectuer leur tracé, s'obtiennent en remarquant que la base EF de chacun des EF qui composent l'espace EF produit E ci-dessus est nodale, ainsi, soit F un des espaces composant l'espace produit E et $u \in F$ alors pour un *ddl* noté d de l'espace F on obtient la composante u_d de u liée à ce *ddl* par $u(d) = u_d$. Le cas d'une base non nodale est traité au second exemple avec l'utilisation de la fonction `inv_interp_EF_ddl` pour le passage des valeurs aux *ddl* d'une fonction à la valeur de ses composantes dans la base EF de son espace EF d'appartenance.

Les figures 1 à 5 obtenues dans cet exemple représentent les fonctions f_i , elle sont obtenues par la commande :

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1 :5], 4) ;
```

Le chiffre 4 indique que les triangles du maillage sont subdivisés en $4^2 = 16$ triangles. Les figures 6 à 10 représentent respectivement les fonctions f_1, f_2, f_3 et le champ de vecteur (f_2, f_3) , elles sont obtenues par la commande :

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1,2,3], 4, 6, [2,3]) ;
```

Enfin la figure 10 est la représentation de la fonction f_1 obtenue par la commande

```
[X2, Y2, T2, U, P, Q, Qx, Qy] = interp_EF(donnees, Uc, 0, [1], ...  
4, 10) ;
```

où `interp_EF` effectue les mêmes tracés que `trac_solEc` et permet aussi, sur chaque triangle, le calcul de polynômes d'interpolation des fonctions représentées par les cellules de Uc . Le *script* du fichier de l'exemple 1 est le suivant.

C :/EF/exemples/trace_graphe_interp/exemple_trac_solEc.m

```
%Fichier EF/exemples/trace_graphe_interp/exemple_trac_solEc.m
```

```
%Utilisation de trac_solEc : traces de fonctions de
```

```
% l'espace E = E1 x E2 x E3 x E4 x E5 avec
```

```
%      E1 = P1 continu = F1
```

```
%      E2 = P2 continu = F2
```

```
%      E3 = P2 continu = F2
```

```
%      E4 = P5 = F3
```

```
%      E5 = P3 discontinu = F4
```

```
clear ;
```

```
close all ;
```

```
%Choix des types d'espaces : F1, F2, F3, F4 :
```

```
D.E = [1,2,5,103] ;
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

```
%Choix de l'espace produit E defini par
%E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Calcul du maillage, variables T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;

%Dont la frontiere, variable Front :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Choix des formules d'integration (pour EF_Prep) :
D.intT = 10 ;
D.intA = 6 ;

%Derivations (pour EF_Prep) :
D.der = [0,1,2] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;
donnees.nom_mail = ...
    'C :/EF/trace_graphe_interp/fich_trac_solEc/f_mail' ;
donnees.nom_prep = ...
    'C :/EF/trace_graphe_interp/fich_trac_solEc/f_prep' ;

%Creation du fichier de maillage
save(donnees.nom_mail,'T','X','Y','Front','Noeu')
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/trac_solEc

```
%Calcul du fichier relatif aux espaces d'EF et aux formules
% d'integration :
EF_Prep(donnees);

%Chargement des coordonnees des DDL :
load(donnees.nom_prep,'Xc','Yc','SL_Dom','Arc');

%Initialisation de diverses valeurs de Uc :
%-----%
%Valeur aux DDL du P1 et composantes P1 de la fonction x + y :
Uc{1} = Xc{1}' + Yc{1}';

%Valeur aux DDL du P2 et composantes P2 de la fonction -y,
% Uc est mis sous forme de vecteur colonne :
Uc{2,1} = -Yc{2}';

%Valeur aux DDL du P2 et composantes P2 de la fonction x :
Uc{3} = Xc{2}';

%Valeur aux DDL du P5 de la fonction exp(x*y) et composantes P5
% de son interpolate P5 :
Uc{4} = exp(Xc{3}'.*Yc{3}');

%Valeur aux DDL du P3d de la fonction x^4 + y^4 et composantes P3d
% de son interpolate P3d :
Uc{5} = Xc{4}'.^4 + Yc{4}'.^4;

%Trace :
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1 :5], 4);
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1,2,3], 4, 6, [2,3]);

%Utilisation de interp_EF :
[X2, Y2, T2, U, P, Q, Qx, Qy] = interp_EF(donnees, Uc, 0, [1], ...
                                            4, 10);
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

```
title('Utilisation de interp\_EF sur x + y');
```

Exemple 2.

La commande est

```
cd C :/EF/exemples/trace_graphe_interp, exemple2_trac_solEc
```

L'espace EF produit E est ici donné par

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

qui traite le cas de l'espace d'EF $P_1 \oplus 3 \text{ bulles } P_4$.

On trace les composantes de la fonction $f = I_E(g)$ où g est la fonction

$$g = (x + y, -y, x, e^{xy}, x^4 + y^4),$$

et $I_E(g)$ la fonction de E qui interpole g aux *ddl* de E . On a encore

$$\begin{aligned} f_1(x, y) &= x + y, f_2(x, y) = -y, f_3(x, y) = x, f_4(x, y) = I_{1,4}(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4), \end{aligned}$$

où pour une fonction f continue $I_{1,4}(f)$ est la fonction qui interpole f aux *ddl* du $P_{1,4}^c$ et $I_3(f)$ est la fonction du P_3^d qui interpole f aux *ddl* du P_3^d . Dans cet exemple pour calculer les composantes de f_4 dans la base EF de l'espace $P_{1,4}^c$ on utilise la fonction `inv_interp_EF_ddl` qui résout le problème linéaire d'interpolation.

Dans le fichier `exemple`, on obtient les composantes de f_4 par les commandes :

```
Uc{4} = exp(Xc{3}'.*Yc{3}')
```

```
Ub = inv_interp_EF_ddl(donnees, Uc, 0, 4, X, Y, T, Xc, Yc, Tc);
```

```
Uc{4} = Ub{4};
```

La figure 1 obtenue par cet exemple représente les fonctions f_i dans un tableau de 5 graphes, elle est obtenue avec les commandes :

```
nfig.n = 1;
```

```
nfig.t = [2,3];
```

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1 :5], 4, nfig, ...  
                             [], X, Y, T, Tc);
```

La variable de valeur 4 indique que l'on traite la quatrième composante de Uc . La figure 2 représente les fonctions f_1, f_2, f_3 et les champs de vecteur $(f_2, f_3), (f_1, f_2)$ dans un tableau de 5 graphes. Cette figure est obtenue par les commandes :

```
nfig.n = 2;
```

```
nfig.t = [2,3];
```

```
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1,2,3], 4, nfig, ...
```

```
[2,3;1,2], X, Y, T, Tc);
```

Un test est réalisé pour vérifier que la composition de la fonction `interp_EF_ddl`, qui calcule la valeur d'une fonction aux *ddl* d'un espace EF produit *E* en fonction de la valeur de ses composantes dans la base EF, avec la fonction `inv_interp_EF_ddl`, qui inversement calcule les composantes de la fonction à partir de ses valeurs aux *ddl*, vaut bien l'identité :

```
U = interp_EF_ddl(donnees, Uc, 0, [1 :5], X, Y, T, Xc, Yc, Tc);
Uc2 = inv_interp_EF_ddl(donnees, U, 0, [1 :5], X, Y, T, Xc, ...
                        Yc, Tc);

for i=1 :5,
    disp(max(abs(Uc{i} - Uc2{i})));
end
```

Enfin la figure 3 est obtenue avec `interp_EF` de manière identique à la figure 10 de l'exemple 1. Le *script* du fichier de l'exemple 2 est le suivant.

C :/EF/exemples/trace_graphe_interp/exemple2_trac_solEc.m

```
%Fichier EF/exemples/trace_graphe_interp/exemple2_trac_solEc.m
%Utilisation de trac_solEc
%  Memes espaces que dans exemple_trac_sol, a l'exception du P5
%  qui est remplace par du P1 + 3 bulles P4, pour donner un
%  exemple d'un espace EF avec des fonctions bulles, avec
%  traces de fonctions de
%  l'espace E = E1 x E2 x E3 x E4 x E5 avec
%      E1 = P1 continu = F1
%      E2 = P2 continu = F2
%      E3 = P2 continu = F2
%      E4 = P1 + 3 bulle P4 continu = F3
%      E5 = P3 discontinu = F4

clear;
close all;

%Choix des types d'espaces EF : F1, F2, F3 (bulles), F4 :
D.E = [1,2,14,103];
```

4.3. DESCRIPTION DES FONCTIONS FONCTION trace_graphe_interp/trac_solEc

```
%Choix de l'espace produit E defini par
%   E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Calcul du maillage, variables T, X, Y ;
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;

%Dont la frontiere, variable Front (facultatif pour le
%   calcul des DDL) :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;

%Chargement des coordonnees des DDL (on peut mettre Front
%   ou []) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Initialisation de diverses valeurs de Uc :
%-----%
%Valeur aux DDL du P1 et composantes P1 de la fonction x + y :
Uc{1} = Xc{1}' + Yc{1}' ;

%Valeur aux DDL du P2 et composantes P2 de la fonction -y,
%   Uc est mis sous forme de vecteur colonne :
Uc{2,1} = -Yc{2}' ;

%Valeur aux DDL du P2 et composantes P2 de la fonction x :
```

```

Uc{3} = Xc{2}';

%Valeur aux DDL du P1 + 3 bulles P4 de la fonction exp(x*y) :
Uc{4} = exp(Xc{3}'.*Yc{3}');

%Composantes P1 + 3 bulles P4 de l'interpole aux DDL du
% P1 + 3 bulles P4 de la fonction exp(x*y) :
Ub = inv_interp_EF_ddl(donnees, Uc, 0, 4, X, Y, T, Xc, Yc, Tc);
Uc{4} = Ub{4};

%Valeur aux DDL du P3d de la fonction x^4 + y^4 et composantes P3d
% de son interpole P3d :
Uc{5} = Xc{4}'.^4 + Yc{4}'.^4;

%Tableau 2x3 de figures :
nfig.n = 1;
nfig.t = [2,3];

%Trace :
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1 :5], 4, nfig, ...
                             [], X, Y, T, Tc);

%Tableau 2x1 de figures :
nfig.n = 2;
nfig.t = [2,3];

%Trace de champs de vecteurs :
[X2, Y2, T2, U] = trac_solEc(donnees, Uc, 0, [1,2,3], 4, nfig, ...
                             [2,3;1,2], X, Y, T, Tc);

%Test d'interpolation aux ddl a partir des composantes dans la base
% EF et fonction inverse : Uc2 doit redonner Uc a l'erreur
% machine pres :
U = interp_EF_ddl(donnees, Uc, 0, [1 :5], X, Y, T, Xc, Yc, Tc);

```


4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

```

Uc2 = inv_interp_EF_ddl(donnees, U, 0, [1 :5], X, Y, T, Xc, ...
                        Yc, Tc) ;

disp(['Verification : on passe des composantes aux ddl aux' ...
      'valeurs en ces ddl et on revient aux composantes']);

for i=1 :5,
    disp(max(abs(Uc{i} - Uc2{i})));
end

%Utilisation de interp_EF :
[X2, Y2, T2, U, P, Q, Qx, Qy] = interp_EF(donnees, Uc, 0, [1], ...
                                           4, 3, [], X, Y, T, Tc) ;

title('Utilisation de interp\_EF sur x + y') ;

```

Algorithme :

`EF/trace_graphe_interp/trac_solEc.m`

On remet ici la partie déclarative de la fonction :

```

function [X2, Y2, T2, U, varargout] = trac_solEc(donnees, Uc, ...
                                                der, r, M, varargin) ;

```

- Tests sur les variables et gestions des variables optionnelles, les variables `nfig`, `vec`, `X`, `Y`, `T`, `Tc` sont alors initialisées par chargement, calcul ou variables d'entrée. Initialisation de `U` à la taille de `Uc` :

```

for i = 1 :length(Uc)
    U{i,1} = [] ;
end ;

```

- `rep(i)` ci-dessous est le numéro d'espace d'EF du *i*-ème espace d'EF E_i de l'espace EF produit E .

```

D = donnees.D ;
rep = D.E(D.f) ;

```

- Nombre de sommets lorsque l'on raffine un triangle en sous-triangles (l'utilisation de `round` est ici facultative) :

```

MX = round((M + 1)*(M + 2)/2) ;

```

- Maillage en M^2 sous-triangles égaux d'un triangle K . Les vecteurs `L1`, `L2` et `L3` calculées dans la commande ci-après sont les coordonnées barycentriques, par rapport à K , des sommets du maillage. La valeur de ces coordonnées barycentriques est intrinsèque et ne dépend pas du triangle. Un exemple de triangle raffiné avec $M = 4$ est donné à la figure 4.2-1 du Paragraphe 4.2, sur le triangle

de référence $\hat{K}$ de sommets de coordonnées en $(0.5, 1)$, $(0, 0)$, $(1, 0)$. Les autres variables calculées sont TK le tableau des numéros de sommets et NK le vecteur des numéros de sommets qui vont de 1 à $(M + 1) * (M + 2)/2$:

```
[TK, NK, L1, L2, L3] = maillage_KR(M) ;
```

Rappel. Les triangles du sous-maillage sont numérotés en partant de celui qui contient le sommet a_1 puis en parcourant les lignes successives de triangles en allant de l'arête $[a_1, a_2]$ à l'arête $[a_2, a_3]$ (cf. la figure 4.2-1 du Paragraphe 4.2).

- Calcul des variables `cas`, `Deg1` et `Deg2` suivantes : `cas` est la liste des numéros des espaces d'EF programmés. L'espace numéro `cas(k)` est alors sur chaque triangle K l'espace $\mathcal{P}_{i,j}(K)$ du Paragraphe 1.4 avec $i = \text{Deg1}(k)$ et $j = \text{Deg2}(k)$:

```
U9 = 1 :9 ;
```

```
U10 = 0 :9 ;
```

```
Un10 = ones(1,10) ;
```

```
%Cas continu espaces 1 a 99
```

```
cas = [U9,10 :99] ;
```

```
Deg1 = [U9,Un10,2*Un10,3*Un10,4*Un10,5*Un10, ...
```

```
6*Un10,7*Un10,8*Un10,9*Un10] ;
```

```
Deg2 = [U9,U10,U10,U10,U10,U10,U10,U10,U10,U10] ;
```

```
%ajout du cas discontinu : espaces numero 100 a 199
```

```
cas = [cas, 100, cas + 100] ;
```

```
Deg1 = [Deg1, 0, Deg1] ;
```

```
Deg2 = [Deg2, 0, Deg2] ;
```

- Calcul de la variable `cas2` correspond aux EF créés par l'utilisateur, ici les espaces d'EF P_{10} -continu et P_{10} -discontinu de numéros respectifs 200 et 300. Pour la création de nouveaux éléments finis dans le code on pourra aussi utiliser la fonction `trac_solEc2` qui ne nécessite pas de modifications lors de ce type d'ajout. On pose alors :

```
cas2 = [200,300] ;
```

avec pour $\mathcal{P}_{i,j}(K)$, les valeurs de $i = j = 10$ ci-après.

- La numérotation globale choisie pour les sommets du sous-maillage est la suivante : sur le triangle k du maillage les numéros de sommets vont de $1 + (k - 1)MX$ à kMX . Nd est la liste des numéros de sommets du triangle courant dans cette numérotation globale. Initialisation de Nd pour le premier triangle du maillage avant la boucle sur l'ensemble des triangles (*Rappel* : $NK = 1, \dots, (M + 2)(M + 1)/2$) :

```
Nd = NK ;
```

- Boucle en k sur les triangles :

```
for k=1 :size(T,1)%Nombre de triangles
```

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc`

- Calcul des coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) des sommets du triangle courant numéro k (pour le maillage EF) et de la variable $\mathbf{de} = (a_1, b_1, c_1) \cdot (b_2 - c_2, -a_2 + c_2, -b_2 + a_2)$ (le symbole $\cdot$ désigne ici le produit scalaire) à partir des coordonnées des sommets du triangle courant donnés par M1 et M2 ci-dessous :

```
M1 = X(T(k,1 :3)) ;
M2 = Y(T(k,1 :3)) ;
a1 = M1(1) ;
b1 = M1(2) ;
c1 = M1(3) ;
a2 = M2(1) ;
b2 = M2(2) ;
c2 = M2(3) ;
de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2] ' ;
```

- Calcul de X1 et Y1 les coordonnées cartésiennes des sommets du sous-maillage du triangle courant :

```
[X1,Y1] = coord_maillage_KB(L1, L2, L3, M1, M2) ;
```

- Passage à la numérotation globale, X2 et Y2 sont les coordonnées des sommets du maillage du domaine raffiné de numéros allant de $1 + (k - 1) * MX$ à $k * MX$:

```
X2(1 + (k-1)*MX : k*MX) = X1 ;
Y2(1 + (k-1)*MX : k*MX) = Y1 ;
```

- Numérotation globale des sommets du triangle, T2 est la structure des sommets du maillage du domaine raffiné :

```
T2((1 + (k-1)*M^2 : k*M^2), 1 :3) = TK + (k-1)*MX ;
```

• Boucle en i sur les composantes de l'espace EF produit E :

```
for i=1 :length(rep)
    if any(i==[r,vec(:)'])%Si i appartient a r ou vec faire :
```

- Vecteur des ddl du i -ème espace de l'espace EF produit E portés par le triangle numéro k :

```
TcIK = Tc{D.f(i)}(k, :) ;
V = Uc{i} ;
```

- Détermination en fonction de $\mathbf{rep}(i)$ des termes $\mathbf{deg1}$ et $\mathbf{deg2}$ qui définissent l'espace $\mathcal{P}_{deg1,deg2}(K)$ du Paragraphe 1.4 :

```
e = rep(i) ;           %e : numero de l'epace EF courant
I = find(e==cas) ;      %si non vide : famille 1
I2 = find(e==cas2) ;    %si non vide : famille 2
d = der ;               %derivation
```

- Valeur de la dérivée indiquée par `d` des fonctions de base de l'espace d'EF numéro `rep(i)` défini par `deg1` et `deg2` (e ci-dessous est facultatif) aux points de coordonnées barycentriques `L1`, `L2` et `L3` sur le triangle courant de paramètres `a1`, `a2`, `b1`, `b2`, `c1`, `c2`, de :

```

if ~isempty(I)%Si famille 1 :
    deg1 = Deg1(I);%Les fonctions de bases
                %sont definies par :
    deg2 = Deg2(I);%degres : "polynomes d'aretes"
                %et "bulles"
    z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
                L1, L2, L3, deg1, deg2);
elseif ~isempty(I2)%Si famille 2 : deg1 = deg2 = 10
    z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
                L1, L2, L3, 10, 10);
end

```

$U\{i\}$: Valeur de la i -ème composante de la fonction aux sommets du maillage du triangle :

```
U{i,1}(Nd,1) = sum(diag(V(TcIK))*z)';
```

```
end;%Fin du si
```

```
end;%Fin de la boucle en j sur les composantes d'espace.
```

- Incrémentation des numéros des sommets du triangle pour le prochain triangle (*Remarque* : on peut aussi poser directement $Nd = 1 + (k - 1) * MX : k * MX$) :

```
Nd = Nd + MX;
```

```
end;%Fin de la boucle en k sur les triangles
```

- Gestion des variables de sortie optionnelles `Tc`, `X`, `Y`, `T`.
- Tracés sauf si `nfig` est de type `CHAR ARRAY`.

Fonction :

```
function [X2, Y2, T2, U, varargout] = trac_solEc2(donnees, Uc, ...
                                                    der, r, M, varargin);
```

Cette fonction est identique à `trac_solEc`, seul l'algorithme est modifié de la manière suivante. L'algorithme diffère par l'utilisation de la fonction `F_base_T` à la place de la fonction `F_base`.

Ainsi dans `trac_solEc2`, le calcul des variables `Deg1`, `Deg2`, `cas` et `cas2` est supprimé.

Ensuite, la boucle décrite dans l'algorithme précédent de la fonction `trac_solEc` :

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/trac_solEc2`

```

for i = 1 :length(rep)
    if any(i==[r,vec( :)]')
        TcIK = Tc{D.f(i)}(k, :) ;
        V = Uc{i} ;
        e = rep(i) ;
        I = find(e==cas) ;
        I2 = find(e==cas2) ;
        if ~isempty(I)
            deg1 = Deg1(I) ;
            deg2 = Deg2(I) ;
            d = der ;%
            z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
                        L1, L2, L3, deg1, deg2) ;
        elseif ~isempty(I2)
            z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
                        L1, L2, L3, 10, 10) ;
        end
        U{i,1}(Nd,1) = sum(diag(V(TcIK))*z,1)' ;
    end ;
end ;
devient
for i=1 :length(rep)
    if any(i==[r,vec( :)]')
        TcIK = Tc{D.f(i)}(k, :) ;
        V = Uc{i} ;
        e = rep(i) ;
        d = der ;
        z = F_base_T(e, d, [], [], a1, a2, b1, b2, c1, ...
                        c2, de, [], L1, L2, L3) ;
        U{i,1}(Nd,1) = sum(diag(V(TcIK))*z,1)' ;
    end ;
end ;

```

qui, est la seconde modification effectuée dans `trac_solEc2` par rapport à `trac_solEc`, le reste de la fonction est identique. Ainsi, si l'on ajoute une espace

d'EF dans le code, en prenant en compte de cet ajout dans `F_base_T`, on n'aura pas à modifier la fonction `trac_solEc2`.

On a une situation identique pour la fonction `trac_solEd` avec `trac_solEd2` et la fonction `interp_EF` avec la fonction `interp_EF2`.

On a aussi ce cas pour la fonction `interp_EF_ddl` avec `interp_EF_ddl2` et pour la fonction `inv_interp_EF_ddl` avec `inv_interp_EF_ddl2` avec en plus pour ces deux cas une modification du calcul des coordonnées barycentriques des *ddl* en utilisant une inversion de système.

Fonction :

```
function [X2, Y2, T2, U, varargout] = trac_solEd(donnees, Uc, ...
                                                der, r, M, varargin);
```

Même syntaxe que `trac_solEc`, seul l'aspect des surfaces diffère. N'utilise pas `pdesurf` de `pdetool`.

Fonction :

```
function [X2, Y2, T2, U, varargout] = trac_solEd2(donnees, Uc, ...
                                                der, r, M, varargin);
```

Cette fonction est identique à `trac_solEd`, seul l'algorithme est modifié de la même manière que `trac_solEc2` relativement à `trac_solEc`, voir ces fonctions.

Fonction :

```
function [X2, Y2, T2, U, varargout] = interp_EF(donnees, Uc, ...
                                                der, r, M, varargin) ;
```

Calcul de polynômes d'interpolation sur chaque triangle des fonctions de l'espace EF produit E , et tracé de ces polynômes. Excepté pour les variables optionnelles de sortie la syntaxe est de cette fonction est la même que celle de `trac_solEc`. Pour cette fonction la réunion des indices de `r` et `vec = varargin{2}` doit être non vide.

Les commandes

```
[X2, Y2, T2, U, P, Q, Qx, Qy] = ...
                                interp_EF(donnees, Uc, der, r, M, ...)
```

ou

```
[X2, Y2, T2, U, P, Q, Qx, Qy, Tc] = ...
                                interp_EF(donnees, Uc, der, r, M, ...)
```

ou

```
[X2, Y2, T2, U, P, Q, Qx, Qy, Tc, X, Y, T] = ...
                                interp_EF(donnees, Uc, der, r, M, ...)
```

effectuent les mêmes tracés et interpolations que

```
[X2, Y2, T2, U] = interp_EF(donnees, Uc, der, r, M, ...)
```

et calcule de plus des polynômes d'interpolation de degré au plus M représentés par les variables `P`, `Q`, `Qx`, `Qy` décrites ci-après (description des variables de sortie). Les autres variables optionnelles `X`, `Y`, `T` sont les variables de maillage et `Tc` le tableau des *ddl*.

Le polynôme d'interpolation de la fonction f_i représentée par $Uc\{i\}$ sur le triangle numéro k est le polynôme $p^{i,k}$ qui interpole f_i aux $(M+1)(M+2)/2$ points du maillage en M^2 sous-triangles égaux vu à la fonction `trac_solEc`. Ce polynôme est représenté de manière équivalente par $P\{i\}(k,:)$ (vecteur ligne) ou $Q\{i\}\{k\}$ (matrice). Le calcul d'interpolation et le tracé sont effectués à partir de $p^{i,k}$ ce qui donne des résultats identiques à ceux donnés par `trac_solEc` à la précision machine près et l'interpolation est exacte aux sommets du sous-maillage.

Pour `der = 1` ou `2`, le calcul d'interpolation et le tracé se font avec $p^{i,k}_{,x}$ ou $p^{i,k}_{,y}$. C'est donc la dérivée du polynôme d'interpolation qui est prise en compte et non la dérivée de f_i . Ainsi les résultats diffèrent *a priori* de ceux obtenus par `trac_solEc` si le degré de $p^{i,k}$ est inférieur à celui de f_i .

Variables d'entrée :

Ce sont les mêmes que celles de `trac_solEc`, la variable `M` aussi est de plus le degré des polynômes d'interpolation comme décrit ci-après. Comme vu ci-dessus, la réunion des indices de `r` et `vec = varargin{2}` doit être non vide.

Variables de sortie :

Les variables non optionnelles sont les mêmes que dans `trac_solEc`.

Variables de sortie optionnelles :

P : CELL ARRAY. On suppose que l'on calcule le polynôme d'interpolation de chaque fonction représentée par $\text{Uc}\{r_1\}, \dots, \text{Uc}\{r_p\}$ avec $r_1 < \dots < r_i < \dots < r_p$ donné. P est un vecteur ligne de longueur m_r .

- Si $i \in \{r_1, \dots, r_p\}$, $\text{P}\{i\}(k, :)$ est le vecteur ligne de type **DOUBLE ARRAY** des coefficients du polynôme $p^{i,k}$ de degré inférieur ou égal à M qui interpole la fonction représentée par $\text{Uc}\{i\}$ aux $(M+1)(M+2)/2$ sommets des triangles obtenus par division en M^2 triangles égaux du triangle numéro k du maillage (*cf.* la Figure 4.2-1, Paragraphe 4.2). Les coefficients sont les composantes de $p^{i,k}$ dans la base suivante :

$$1, x, x^2, \dots, x^M, y, xy, x^2y, \dots, x^{M-1}y, \dots, y^{M-1}, xy^{M-1}, y^M.$$

- Sinon, on a $\text{P}\{i\} = []$.

Q : CELL ARRAY. Vecteur ligne de même longueur que P, on reprend les notations utilisées pour la variable P.

- Si $i \in \{r_1, \dots, r_p\}$, $\text{Q}\{i\}\{k\}$ est une matrice carrée de type **DOUBLE ARRAY** de taille $M+1$ qui représente le polynôme $p^{i,k}$ donné dans la définition de la variable P ci-dessus. Pour le triangle numéro k , le coefficient de $x^{m-1}y^{n-1}$ de $p^{i,k}$, $2 \leq m+n \leq M+2$, est donné par le terme (m, n) de $\text{Q}\{i\}\{k\}$. Si $m+n > M+2$ on pose $\text{Q}\{i\}\{k\} = 0$ (les termes sous l'anti-diagonale de $\text{Q}\{i\}\{k\}$ sont nuls). La valeur de $p^{i,k}$ en un point (x, y) est donnée par

$$p^{i,k}(x, y) = \text{sum}(\text{sum}(\text{Q}\{i\}\{k\} .* (\text{x}.^{[0:M]}' * \text{y}.^{[0:M]}))).$$

- Sinon, on a $\text{Q}\{i\} = []$.

QX, QY : CELL ARRAY. Analogues de **Q** : $\text{QX}\{i\}\{k\}$ et $\text{QY}\{i\}\{k\}$ sont les deux matrices carrées de type **DOUBLE ARRAY** de taille $M+1$ qui représentent respectivement les polynômes $p_{,x}^{i,k}$ et $p_{,y}^{i,k}$ de la définition de P ci-dessus avec la même représentation que Q ci-dessus.

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type **DOUBLE ARRAY**. La matrice $\text{Tc}\{i\}$ décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

X, Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/interp_EF`

Exemple :

Un exemple d'utilisation est donné dans le fichier *script exemple_trac_solEc.m* donné dans la description de la fonction `trac_solEc` avec dans ce fichier la commande

`%Utilisation de interp_EF :`

```
[X2, Y2, T2, U, P, Q, Qx, Qy] = interp_EF(donnees, Uc, 0, [1], ...
                                          4, 10) ;
```

pour le tracé et l'interpolation de la fonction f_1 définie après $f_1(x, y) = x + y$ appartenant à l'espace d'EF P_1^c , premier espace de l'espace EF produit E utilisé dans cet exemple.

Le polynôme de degré inférieur ou égal à 4 qui interpole f_1 sur chaque triangle est donc $p(x, y) = f(x, y) = x + y$. On trouve ainsi en sortie :

$$P\{1\}(i, :) = (0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0), 1 \leq i \leq 8 \text{ (nombre de triangles)}$$

$$Q\{1\}\{i\} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, 1 \leq i \leq 8$$

$$Qx\{1\}\{i\} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, Qy\{1\}\{i\} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} 1 \leq i \leq 15.$$

Algorithme :

`EF/trace_graphe_interp/interp_EF`

On remet ici la partie déclarative de la fonction :

```
function [X2, Y2, T2, U, varargout] = interp_EF(donnees, Uc, ...
                                                der, r, M, varargin) ;
```

L'algorithme ci-dessous est identique à celui de la fonction `trac_solEc` excepté la partie mise entre deux lignes.

- Tests sur les variables et gestions des variables optionnelles, les variables `nfig`, `vec`, `X`, `Y`, `T`, `Tc` sont alors initialisées par chargement, calcul ou variables d'entrée.

`%Initialisation de U a la taille de Uc`

```
for i = 1 :length(Uc)
    U{i,1} = [] ;
```

`end ;`

- `rep(i)` ci-dessous est le numéro d'espace d'EF du i -ème espace d'EF E_i de l'espace EF produit E .

`D = donnees.D ;`

`rep = D.E(D.f) ;`

- Nombre de sommets du triangle raffiné (`round` facultatif) :

`MX = round((M + 1)*(M + 2)/2) ;`

- Maillage en M^2 sous-triangles du triangle de référence $\hat{K}$ de sommets de coordonnées $(0.5, 1)$, $(0, 0)$, $(1, 0)$, les vecteurs `L1`, `L2` et `L3` sont les coordonnées barycentriques par rapport à $\hat{K}$, des sommets du maillage de sous-triangles, `TK` le tableau des numéros de sommets et `NK` le vecteur des numéros de sommets qui vont de 1 à $(M + 1) * (M + 2) / 2$ (un exemple de triangle raffiné avec $M = 4$ est donné à la figure 4.2-1 du Paragraphe 4.2) :

`[TK, NK, L1, L2, L3] = maillage_KR(M) ;`

Rappel. Les triangles du sous-maillage sont numérotés en partant du sous-triangle auquel appartient le sommet a_1 puis en parcourant les 3 sous-triangles placés sous ce triangle en allant de l'arête $[a_1, a_2]$ vers l'arête $[a_1, a_3]$, etc, en terminant par les triangles adjacents à l'arête $[a_2, a_3]$ (*cf.* figure sur la figure 4.2-1 du Paragraphe 4.2).

- Calcul des variables `cas`, `Deg1` et `Deg2` suivantes : `cas` est la liste des numéros des espaces d'EF programmés. L'espace numéro `cas(k)` est alors sur chaque triangle K l'espace $\mathcal{P}_{i,j}(K)$ du Paragraphe 1.4 avec $i = \text{Deg1}(k)$, $j = \text{Deg2}(k)$:

`U9 = 1 : 9 ;`

`U10 = 0 : 9 ;`

`Un10 = ones(1,10) ;`

`%Cas continu espaces 1 a 99`

`cas = [U9,10 : 99] ;`

`Deg1 = [U9,Un10,2*Un10,3*Un10,4*Un10,5*Un10, ...`

`6*Un10,7*Un10,8*Un10,9*Un10] ;`

`Deg2 = [U9,U10,U10,U10,U10,U10,U10,U10,U10,U10] ;`

`%ajout du cas discontinu : espaces numero 100 a 199`

`cas = [cas, 100, cas + 100] ;`

`Deg1 = [Deg1, 0, Deg1] ;`

`Deg2 = [Deg2, 0, Deg2] ;`

- Calcul de la variable `cas2` correspond aux EF créés par l'utilisateur, ici les espaces d'EF P_{10} -continu et P_{10} -discontinu de numéros respectifs 200 et 300. Pour la création de nouveaux éléments finis dans le code on pourra aussi utiliser la

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/interp_EF`

fonction `interp_EF2` qui ne nécessite pas de modifications pour ce type d'ajout. On pose alors :

`cas2 = [200,300] ;`

avec pour $\mathcal{P}_{i,j}(K)$, les valeurs de $i = j = 10$ ci-après.

- La numérotation globale choisie pour les sommets du sous-maillage est la suivante : sur le triangle k du maillage les numéros des sommets vont de $1 + (k-1)MX$ à kMX . La variable `Nd` est la liste des numéros des sommets du triangle courant dans cette numérotation globale, initialisation de `Nd` pour le premier triangle du maillage (*rappel* : $NK = (1, \dots, (M+2)(M+1)/2)$) :

`Nd = NK ;`

- Boucle en k sur les triangles du maillage EF :

`for k=1 :size(T,1)%Nombre de triangles`

- Calcul des coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) des sommets du triangle courant numéro k (pour le maillage EF) et de la variable $\mathbf{de} = (a_1, b_1, c_1) \cdot (b_2 - c_2, -a_2 + c_2, -b_2 + a_2)$ (produit scalaire) à partir des coordonnées des sommets du triangle courant donnés par `M1` et `M2` ci-dessous :

`M1 = X(T(k,1 :3)) ;`

`M2 = Y(T(k,1 :3)) ;`

`a1 = M1(1) ;`

`b1 = M1(2) ;`

`c1 = M1(3) ;`

`a2 = M2(1) ;`

`b2 = M2(2) ;`

`c2 = M2(3) ;`

`de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;`

- Calcul de `X1` et `Y1` les coordonnées cartésiennes des sommets du sous-maillage du triangle courant :

`[X1,Y1] = coord_maillage_KB(L1,L2,L3,M1,M2) ;`

- Passage à la numérotation globale, `X2` et `Y2` sont les coordonnées des sommets du maillage du domaine raffiné de numéros allant de $1 + (k-1) * MX$ à $k * MX$:

`X2(1 + (k-1)*MX : k*MX) = X1 ;`

`Y2(1 + (k-1)*MX : k*MX) = Y1 ;`

- Numérotation globale des sommets du triangle, `T2` est la structure des sommets du maillage du domaine raffiné :

`T2((1 + (k-1)*M^2 :k*M^2), 1 :3) = TK + (k-1)*MX ;`

- Boucle en i sur les composantes de l'espace EF produit E :

```
for i=1 :length(rep)
    if any(i==[r,vec( :)]')%Si i appartient a r ou vec faire :
        - Vecteur des ddl du i-ème espace de l'espace EF produit E portés par le triangle
        numéro k :
```

```
TcIK = Tc{D.f(i)}(k, :) ;
```

```
V = Uc{i} ;
```

```
- Détermination en fonction de rep(i) des termes deg1 et deg2 qui définissent
l'espace  $\mathcal{P}_{deg1,deg2}(K)$  du Paragraphe 1.4.
```

```
e = rep(i) ;           %e : numero de l'espace EF courant
```

```
I = find(e==cas) ;     %si non vide : famille 1
```

```
I2 = find(e==cas2) ; %si non vide : famille 2
```

Partie qui change par rapport à `trac_solEc` :

- on calcule les valeurs des fonction de base, la prise en compte de la dérivation se fait avec le polynôme d'interpolation et les polynômes dérivés :

```
d = 0 ;                %derivation
```

Fin de la partie qui change par rapport à `trac_solEc` .

- Valeur de la dérivée indiqué par $d = 0$ des fonctions de base de l'espace d'EF numéro $\text{rep}(i)$ défini par deg1 et deg2 (e ci-dessous est facultatif) aux points de coordonnées barycentriques L1, L2 et L3 sur le triangle courant de paramètres a1, a2, b1, b2, c1, c2, de :

```
if ~isempty(I)%Si famille 1 :
```

```
    deg1 = Deg1(I) ; %Les fonctions de bases
```

```
           %sont definies par :
```

```
    deg2 = Deg2(I) ; %degres : "polynomes d'aretes"
```

```
           %et "bulles"
```

```
    z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
```

```
                L1, L2, L3, deg1, deg2) ;
```

```
elseif ~isempty(I2)%Si famille 2 : deg1 = deg2 = 10
```

```
    z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
```

```
                L1, L2, L3, 10, 10) ;
```

```
end
```

UI : Valeurs de la *i*-ème composante de la fonction à interpoler aux points d'interpolation :

```
UI = sum(diag(V(TcIK))*z,1)' ;
```

Partie qui change par rapport à `trac_solEc` :

Calcul des polynômes

```
%X1,Y1 Coordonnees des sommets du triangle maille
%Calcul des polynomes
[p,q,qx,qy,M] = interp_2D(X1,Y1,UI') ;
P{i}(k, :) = p ;
Q{i}{k} = q ;
Qx{i}{k} = qx ;
Qy{i}{k} = qy ;
z = [] ;
for j=1 :length(X1)
    x1 = X1(j).^[0 :M] ;
    y1 = Y1(j).^[0 :M] ;
    a = x1'*y1 ;
    switch der
    case 0, z(j) = sum(sum(q.*a)) ;
    case 1, z(j) = sum(sum(qx.*a)) ;
    case 2, z(j) = sum(sum(qy.*a)) ;
    end ;
end ;
```

$U\{i\}$: Valeurs, à partir des polynômes, de la *i-ème* composante de la fonction à interpoler aux points d'interpolation :

```
U{i,1}(Nd,1) = z ;
end ;%Fin du si
```

```
end ;%Fin de la boucle en j sur les composantes d'espace.
```

Le résultat du calcul est identique à celui effectué dans `trac_solEc`, on pourra ajouter ici la prise en compte de dérivées d'ordre supérieur.

Fin de la partie qui change par rapport à `trac_solEc` .

- Incrémentation des numéros des sommets du triangle pour le prochain triangle

```
Nd = Nd + MX ;
```

```
end ;%Fin de la boucle en k sur les triangles
```

- Gestion des variables de sortie optionnelles : P, Q, Qx, Qy, Tc, X, Y, T.
- Tracés sauf si `nfig` est de type `CHAR ARRAY`.

Fonction :

```
function [X2, Y2, T2, U, varargout] = interp_EF2(donnees, Uc, ...  
                                                der, r, M, varargin) ;
```

Cette fonction est identique à `interp_EF`, seul l'algorithme est modifié de la même manière que dans `trac_solEc2`, voir cette fonction. Ainsi, si l'on ajoute une espace d'EF dans le code, en prenant en compte de cet ajout dans `F_base_T`, on n'aura pas à modifier la fonction `interp_EF2`.

Fonction :

```
function [X2, Y2, T2, U, varargout] = interp_EF_p(donnees, Uc, ...  
                                                  der, r, M, varargin) ;
```

Syntaxe identique à `interp_EF`, les tracés ne sont pas effectués.

Fonction :

```
function [X2, Y2, T2, U, varargout] = interp_EF_p2(donnees, Uc, ...  
                                                    der, r, M, varargin) ;
```

Syntaxe identique à `interp_EF2`, les tracés ne sont pas effectués.

Fonction :

```
function [U, varargout] = interp_EF_ddl(donnees, Uc, varargin) ;
```

Pour une fonction d'un espace EF produit E , calcule, à partir de ses composantes dans la base choisie dans le code EF, sa valeur aux *ddl* de l'espace EF.

La commande

```
U = interp_EF_ddl(donnees, Uc) ;
```

calcule la valeur, aux *ddl* de l'espace EF produit $E = E_1 \times \dots \times E_i \times \dots \times E_n$ décrit par la variable `donnees`, de la fonction $f = (f_1, \dots, f_i, \dots, f_n) \in E$ représentée par la vecteur colonne de type `CELL ARRAY` `Uc`.

La cellule i de `Uc` est le vecteur colonne des composantes de $f_i \in E_i$ dans la base choisie dans le code EF pour l'espace E_i .

La variable de sortie `U` est un vecteur colonne de type `CELL ARRAY` tel que `U{i}` est le vecteur colonne des valeurs de f_i aux *ddl* de E_i .

La commande

```
U = interp_EF_ddl(donnees, Uc, [], r) ;
```

permet de spécifier par le vecteur ligne ou colonne `r`, les composantes de `Uc` à traiter. Pour `r = []` ce sont toutes les composantes de `Uc` qui sont traitées.

La commande

```
U = interp_EF_ddl(donnees, Uc, der, r) ;
```

permet de plus de donner la valeur des dérivées aux *ddl*. Cette dérivation est indicative lorsque la fonction n'est pas dérivable suivant la direction indiquée par `der`. Sur les *ddl* d'arêtes ou de sommets du maillage, on prend une des dérivées (la dernière rencontrée dans le parcours des triangles). Les valeurs de `der` sont `[]` ou 0 (pas de dérivation), 1 (dérivée par rapport à x) et 2 (dérivée par rapport à y).

La commande

```
U = interp_EF_ddl(donnees, Uc, [], r, X, Y, T) ;
```

permet de calculer les valeurs aux *ddl* de la fonction représentée par `Uc` à partir de `X`, `Y` et `T` et ce sans passer par les fichiers de maillage et de données relatives aux EF. Il n'est alors pas nécessaire que les champs `donnees.nom_mail` et `donnees.nom_prep` soient initialisés. La fonction `interp_EF_ddl` est ici indépendante de ces deux champs.

La commande

```
U = interp_EF_ddl(donnees, Uc, [], r, X, Y, T, Xc, Yc, Tc) ;
```

réalise le même type d'appel que le précédent, mais plus rapide car les variables `Xc`, `Yc` et `Tc` ne sont pas recalculées.

Les commandes du type

```
[U, Xc, Yc, Tc] = interp_EF_ddl(donnees, Uc, ...);
```

permet de récupérer en sortie les 3 variables décrivant les *ddl* et leurs coordonnées.

Variables d'entrée :

`donnees` : STRUCT ARRAY. Variable décrivant les données du problème EF. Pour les appels du type

```
interp_EF_ddl(donnees, Uc) ;
interp_EF_ddl(donnees, Uc, der) ;
interp_EF_ddl(donnees, Uc, der, r) ;
```

les champs à initialiser sont :

```
donnees.D.E,
donnees.D.f,
donnees.nom_prep,
donnees.nom_mail.
```

Et pour les appels du type

```
interp_EF_ddl(donnees, Uc, der, r, X, Y, T, ...)
```

avec appel avec des variables de maillage `X`, `Y`, `T` et éventuellement de `Tc`, les champs à initialiser sont alors :

```
donnees.D.E,
donnees.D.f.
```

`Uc` : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

Variables d'entrée optionnelles :

`der` : DOUBLE ARRAY. Les valeurs de `der` = 0,1,2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

`r` : DOUBLE ARRAY. Variable portant sur les composantes de l'espace EF produit E . Vecteur ligne ou colonne, `r(i)` désigne la composante de la fonction dont on veut les valeurs. La valeur `r = []` donne les valeurs de toutes les composantes.

`X,Y` : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`T` : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`Xc`, `Yc` : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs `Xc{i}` et `Yc{i}` sont les vecteurs des

coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice **Tc**{ i } décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

U : CELL ARRAY. Vecteur colonne de même taille que **Uc**. Si $\mathbf{r} = []$ alors **U**{ i } est le vecteur colonne telle que **U**{ i }(j) est la valeur, au *ddl* numéro j du i -ème espace d'EF de l'espace EF produit E , de la i -ème composante de la fonction (ou, si $\mathbf{der} = 1$ ou 2 , de l'une de ses dérivées qui est alors indicative) représentée par **Uc**. Si $\mathbf{r}$ est un vecteur d'entier alors le calcul n'est effectuée que pour les composante i appartenant à $\mathbf{r}$.

Variables de sortie optionnelles :

Xc, **Yc** : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs **Xc**{ i } et **Yc**{ i } sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice **Tc**{ i } décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Exemple 1 : Le fichier *exemple_interp_EF_ddl.m* ci-dessous utilise la fonction `interp_EF_ddl` sans utilisation du fichier `donnees.nom_prep`, les *ddl* étant calculés par la fonction `ddl`.

Ce fichier vérifie aussi sur les exemples que la composition de la fonction `interp_EF` avec sa fonction réciproque `inv_interp_EF`, qui calcule les composantes d'une fonction de l'espace EF produit E à partir de ses valeurs aux *ddl*, redonne bien l'identité.

Cet exemple est mis en œuvre par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple_interp_EF_ddl,
```

Cette exemple reprend les espaces et les fonctions de l'exemple 2 donné dans la description de la fonction `trac_solEc` avec le fichier *exemple2_trac_solEc.m*. L'espace EF produit E est donné par

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

qui traite le cas de l'espace d'EF $P_1 \oplus 3 \text{ bulles } P_4$, avec le calcul des valeurs aux *ddl* de la fonction $f = (f_1, f_2, f_3, f_4, f_5) \in E$ donnée par $f = I_E(g)$ où g est la fonction

$$g = (x + y, -y, x, e^{xy}, x^4 + y^4),$$

$I_E(g)$ étant la fonction de E qui interpole g aux *ddl* de E . On a encore

$$\begin{aligned} f_1(x, y) &= x + y, f_2(x, y) = -y, f_3(x, y) = x, f_4(x, y) = I_{1,4}(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4), \end{aligned}$$

où, pour une fonction continue f , $I_{1,4}(f)$ est la fonction qui interpole f aux *ddl* du $P_{1,4}^c$ et $I_3(f)$ est la fonction du P_3^d qui interpole f aux *ddl* du P_3^d (f peut alors être continue seulement sur les triangles).

Dans le fichier `exemple`, nous calculons d'abord les composantes de f dans la base EF de E , puis nous appliquons la fonction `interp_EF_ddl` pour calculer les valeurs de f aux *ddl* de E . A titre de vérification nous appliquons la fonction `inv_interp_EF_ddl` au résultat obtenu, ce qui doit redonner les composantes de f .

Dans le fichier `exemple` les composantes de f sont calculées à partir des valeurs de g aux *ddl* de la manière suivante :

- On pose pour $g_1(x, y) = x + y$:

$$\text{Uc}\{1\} = \text{Xc}\{1\}' + \text{Yc}\{1\}' ;$$

$\text{Uc}\{1\}$ représente alors les valeurs aux *ddl* de l'espace d'EF P_1^c de la fonction $f_1(x, y) = x + y$, qui sont aussi les composantes de f_1 car, par définition de l'interpolation sur E , $f_1 \in P_1^c$ et la base est nodale. On procède de manière identique pour $g_2(x, y) = -y$ et $g_3(x, y) = x$ et l'espace d'EF P_2^c .

- On pose ensuite pour $g_4(x, y) = e^{xy}$:

$$\text{Uc}\{4\} = \exp(\text{Xc}\{3\}' \cdot \text{Yc}\{3\}') ;$$

et $\text{Uc}\{4\}$ représente les valeurs aux *ddl* de l'espace d'EF $P_{1,4}^c$ de la fonction f_4 , comme la base n'est pas nodale on utilise ici les commandes

$$\begin{aligned} \text{Ub} &= \text{inv\_interp\_EF\_ddl}(\text{donnees}, \text{Uc}, 0, 4, \text{X}, \text{Y}, \text{T}, \text{Xc}, \text{Yc}, \text{Tc}) ; \\ \text{Uc}\{4\} &= \text{Ub}\{4\} ; \end{aligned}$$

et $\text{Uc}\{4\}$ représente alors les composantes dans l'espace d'EF $P_{1,4}^c$ de la fonction f_4 . On remarque que l'on pourra aussi utiliser `inv_interp_EF_ddl` dans le cas d'une base nodale (*cf.* l'exemple 2 suivant), dans ce cas, valeurs et composantes aux *ddl* sont identiques.

- Enfin, on pose pour $g_5(x, y) = x^4 + y^4$:

$$\text{Uc}\{5\} = \text{Xc}\{4\}' \cdot \wedge^4 + \text{Yc}\{4\}' \cdot \wedge^4 ;$$

$\text{Uc}\{5\}$ représente les valeurs aux *ddl* de l'espace d'EF P_3^d de la fonction f_4 , qui sont aussi ses composantes car la base est nodale.

- La variable Uc représente alors les composantes dans E de f . Les valeurs de $f \in E$ aux *ddl* de E sont alors données en fonction de ses composantes dans E par

$$\text{U} = \text{interp\_EF\_ddl}(\text{donnees}, \text{Uc}, [], [], \text{X}, \text{Y}, \text{T}, \text{Xc}, \text{Yc}, \text{Tc}) ;$$

- Réciproquement on calcule, avec la commande

$Uc2 = \text{inv\_interp\_EF\_ddl}(\text{donnees}, U, [], [], X, Y, T, Xc, Yc, Tc);$
 les composantes de f dans E en fonction des valeurs de f aux ddl de E

- On vérifie que $Uc2\{i\} = U\{i\}$ pour $i = 1, 2, 3, 5$, car dans de cas valeurs aux ddl et valeurs des composantes attachées à ces ddl sont identiques dans le cas d'une base nodale.

- Et enfin on vérifie que $Uc2\{i\} = Uc\{i\}$ pour $1 \leq i \leq 5$ car la composition de interp_EF_ddl avec inv_interp_EF_ddl donne l'identité.

Le fichier *script* est le suivant.

EF/exemples/trace_graphe_interp/exemple_interp_EF_ddl.m (exemple 1)

```
%Fichier EF/exemples/trace_graphe_interp/exemple_interp_EF_ddl.m
%Exemple d'utilisation de INTERP_EF_DDL. On considere un espace
% d'EF E = E1 x E2 x E3 x E4 x E5 et une fonction f de E notee
% f = (f1, f2, f3, f4, f5) de E. On pose :
%
%      E1 = P1 continu = F1,
%      E2 = P2 continu = F2,
%      E3 = P2 continu = F2,
%      E4 = P1 + 3 bulles P4 continu = F3,
%      E5 = P3 discontinu = F4.
%
% Les calculs sont effectues sur la fonction f = I_E(g) qui
% est la fonction de E qui interpole aux DDL de l'espace E
% la fonction g definie par :
%
%      g = (x + y, -y, x, exp(xy), x^4 + y^4).
%
% - On effectue avec INTERP_EF_DDL le calcul des valeurs de f
% aux DDL de l'espace EF en fonction de ses composantes dans E.
%
% - On donne aussi un exemple d'utilisation de la fonction reci-
% proque INV_INTERP_EF_DDL qui calcule les composantes de f dans
% E a partir de ses valeurs aux DLL.

clear ;
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/interp_EF_ddl

```

close all ;

%Choix des types d'espaces EF F1, F2, F3 (bulles) et F4 :
D.E = [1,2,14,103] ;

%Choix de l'espace produit E defini par :
%   E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Definition du maillage et des variables de maillage T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;

%Calcul du terme frontiere Front du maillage qui est ici facultatif
%   pour le calcul des DDL qui suit :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;

%Calcul de la structure des DDL (on peut aussi poser Front = [] ci-
%   dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Calcul de Uc representant la fonction f = I_E(g) :
%-----%
%Valeur aux DDL de E1 = P1 de la fonction f1(x,y) = g1(x,y) =
%   x + y. Ces valeurs sont aussi les composantes sur E1 de f1,
%   car la base est nodale :
```

```
Uc{1} = Xc{1}' + Yc{1}';
```

```
%Valeur aux DDL de E2 = P2 et composantes sur le P2 de la fonction
% f2(x,y) = -y. Uc est mis sous forme de vecteur colonne :
Uc{2,1} = -Yc{2}';
```

```
%Valeur aux DDL de E3 = P2 et composantes sur le P2 de la fonction
% f3(x,y) = x :
Uc{3} = Xc{2}';
```

```
%Valeur aux DDL de E4 = P1 + 3 bulles P4 de la fonction
% g4(x,y) = exp(x*y) :
Uc_val{4,1} = exp(Xc{3}'.*Yc{3}')
```

```
%Calcul des composantes sur E4 de f4(x,y) qui interpole la fonction
% g4(x,y) = exp(x*y) aux DDL de E4. Comme la base de E4, qui est
% un espace d'EF avec fonctions de type bulles, n'est pas nodale,
% les composantes de f4 sur E4 sont obtenues en appliquant la
% fonction INV_INTERP_EF_DDL a Uc_val{4} :
Ub = inv_interp_EF_ddl(donnees, Uc_val, 0, 4, X, Y, T, Xc, Yc, Tc);
Uc{4} = Ub{4};
```

```
%Valeur aux DDL de E5 = P3d de la fonction g5(x,y) = x^4 + y^4. Ces
% valeurs sont aussi les composantes sur E5 de la fonction f5 qui
% interpole g5 aux DDL de E5, car la base est nodale :
Uc{5} = Xc{4}'.^4 + Yc{4}'.^4;
```

```
%Calcul de U donnant les valeurs aux DDL de E de la fonction f de
% E :
%-----%
U = interp_EF_ddl(donnees, Uc, [], [], X, Y, T, Xc, Yc, Tc);
```

```
%Calcul de Uc2 representant les composantes de la fonction f de
% E a partir des valeurs de f aux DDL de E (calcul reciproque) :
```

```
%-----%
Uc2 = inv_interp_EF_ddl(donnees, U, [], [], X, Y, T, Xc, Yc, Tc);

%On verifie que U{i} et Uc{i} sont identiques pour i = 1, 2, 3, 5.
% Ceci vient du fait que la base de Ei est alors nodale :
format compact;
disp(['Ecart entre la valeur aux DDL de la fonction etudiee ', ...
      'et la valeur de ses composantes dans la base EF :']);
for i=1 :5,
    disp(['element numero : ', num2str(D.E(D.f(i)))]);
    disp(max(abs(U{i} - Uc{i})));
end;
%On verifie que Uc et Uc2 sont identiques. Ceci vient du fait que
% Uc2 est obtenu a partir de Uc par composition de INTERP_EF_DDL
% et de sa fonction reciproque INV_INTERP_EF_DDL :
disp(['Ecart entre les composantes calculees en appliquant ' ...
      'a la fonction etudiee interp_EF_ddl puis ', ...
      'inv_interp_EF_ddl, avec ses composantes dans la base EF :']);
for i=1 :5,
    disp(['element numero : ', ...
          num2str(D.E(D.f(i)))]), disp(max(abs(Uc{i} - Uc2{i})));
end;
format;
```

Exemple 2 : Variante de `exemple_interp_EF_ddl` pour la mise en œuvre des dérivations. La commande pour cet exemple est

```
cd C :/EF/exemples/trace_graphe_interp, exemple2_interp_EF_ddl;
```

Cette exemple reprend les espaces et les fonctions de l'exemple 1 précédent avec l'espace EF produit E donné par

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

et calcule, avec la fonction `interp_EF_ddl`, les valeurs aux *ddl* de la dérivée suivant x de la fonction $f = (f_1, f_2, f_3, f_4, f_5) \in E$ donnée par $f = I_E(g)$ avec

$$g = (x + y, -y, x, e^{xy}, x^4 + y^4).$$

On a aussi

$$\begin{aligned} f_1(x, y) &= x + y, \quad f_2(x, y) = -y, \quad f_3(x, y) = x, \quad f_4(x, y) = I_{1,4}(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4). \end{aligned}$$

Ce fichier compare les composantes dans E de $I_E(g_{,x})$ la fonction de E qui interpole $g_{,x}$ aux *ddl* de E avec les composantes de $I_E(f_{,x})$ la fonction de E qui interpole $f_{,x}$ aux *ddl* de E .

Pour $g_{,x}$, le calcul d'interpolation est réalisé à partir de ses valeurs aux *ddl* et utilisation de la fonction `inv_interp_EF_ddl`.

Pour $f_{,x}$, le calcul d'interpolation est réalisé avec la fonction `inv_interp_EF_ddl` à partir des valeurs aux *ddl* de $f_{,x}$ calculées par `interp_EF_ddl`.

Pour $i = 4, 5$, un écart apparaît entre les valeurs de $f_{i,x}$ et $g_{i,x}$ car g_i n'étant pas une fonction de E_i diffère de son interpolé f_i . Cet écart se vérifie ensuite sur les composantes dans E de $I_E(f_{,x})$ et $I_E(g_{,x})$. On note un écart important (1.9889) entre $f_{4,x}$ et $g_{4,x}$.

Il est à noter que $f_{4,x}$ et $f_{5,x}$ ne sont pas *a priori* des polynômes sur Ω (seulement des polynômes par morceaux) et ne sont pas *a priori* dérivables sur les arêtes du maillage. L'utilisation de `interp_EF_ddl` pour la dérivation sera alors utilisé avec précaution.

Le fichier *script* de l'exemple 2 est le suivant.

EF/exemples/trace_graphe_interp/exemple2_interp_EF_ddl.m (exemple 2)

```
%Fichier EF/exemples/trace_graphe_interp/exemple2_interp_EF_ddl.m
%Exemple d'utilisation de INTERP_EF_DDL. On considere un espace
% d'EF E = E1 x E2 x E3 x E4 x E5 et une fonction f de E notee
% f = (f1, f2, f3, f4, f5) de E. On pose :
%
%      E1 = P1 continu = F1,
%      E2 = P2 continu = F2,
%      E3 = P2 continu = F2,
%      E4 = P1 + 3 bulles P4 continu = F3,
%      E5 = P3 discontinu = F4.
%
% Les calculs sont effectues sur la fonction f = I_E(g) qui
% est la fonction de E qui interpole aux DDL de l'espace E
% la fonction g definie par :
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/interp_EF_ddl

```
%
%
%          g = (x + y, -y, x, exp(xy), x^4 + y^4).
%
%
%  On effectue avec INTERP_EF_DDL le calcul des valeurs aux DDL de
%
%
%          f,x avec f = I_E(g) et de g,x
%
%
%  et on compare les resultats obtenus (cf les variables V et V2
%  ci-dessous).
%
%
%  On effectue avec INV_INTERP_EF_DDL le calcul des composantes de
%
%
%          I_E(f,x)  et I_E(g,x)
%
%
%  et on compare les resultats obtenus (cf les variables Vc et Vc2
%  ci-dessous).

clear ;
close all ;

%Choix des types d'espaces EF F1, F2, F3 (bulles) et F4 :
D.E = [1,2,14,103] ;

%Choix de l'espace produit E defini par :
%  E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Definition du maillage et des variables de maillage T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;

%Calcul du terme frontiere Front du maillage qui est ici facultatif
```



```

% pour le calcul des DDL qui suit :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;

%Calcul de la structure des DDL (on peut aussi poser Front = [] ci-
% dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Uc composantes dans E de f = I_E(g) :
%-----%
%a) Valeur aux DDL de la fonction g ;
U{1} = Xc{1}' + Yc{1}' ;
U{2,1} = -Yc{2}' ;
U{3} = Xc{2}' ;
U{4,1} = exp(Xc{3}'.*Yc{3}') ;
U{5} = Xc{4}'.^4 + Yc{4}'.^4 ;
%b) Composantes de f ;
Uc = inv_interp_EF_ddl(donnees, U, 0, [], X, Y, T, Xc, Yc, Tc) ;

%V valeur aux DLL de f,x = (I_E(g)),x :
%-----%
% Derivation approchee si f n'est pas derivable en tout points
V = interp_EF_ddl(donnees, Uc, 1, [], X, Y, T, Xc, Yc, Tc) ;

%Vc composantes dans E de I_E(f,x) :
%-----%
% Vc est obtenue a partir des valeurs aux DDL de f,x :
Vc = inv_interp_EF_ddl(donnees, V, 0, [], X, Y, T, Xc, Yc, Tc) ;

```

```

%V2 valeur aux DLL de g,x, Vc2 composantes dans E de I_E(g,x) :
%-----%
%a) Valeurs aux DDL de g,x :
V2{1} = Xc{1}'.^0 ;
V2{2,1} = 0*Yc{2}' ;
V2{3} = Xc{2}'.^0 ;
V2{4} = Yc{3}'.*exp(Xc{3}'.*Yc{3}');
V2{5} = 4*Xc{4}'.^3 ;
%b) Composantes de I_E(g,x) :
Vc2 = inv_interp_EF_ddl(donnees, V2, 0, [], X, Y, T, Xc, Yc, Tc);

%Affichage comparaisons :
format compact;
%On retrouve les memes resultats lorsque gi est dans Ei :
disp(['Ecart entre les valeurs aux DDL de f,x et g,x :']);
for i=1 :5,
    disp(['element numero : ', num2str(D.E(D.f(i)))]);
    disp(max(abs(V{i} - V2{i})));
end;
%On retrouve les memes resultats lorsque gi est dans Ei :
disp(['Ecart entre les composantes de I_E(f,x) et I_E(g,x) :']);
for i=1 :5,
    disp(['element numero : ', ...
        num2str(D.E(D.f(i)))], disp(max(abs(Vc{i} - Vc2{i})));
end;
format ;

```

Exemple 3 : L'exemple cité ci-dessous, et donné dans la description de la fonction `trac_solEc`, utilise la fonction `trac_solEc`, la fonction `interp_EF` pour donner des polynômes d'interpolation en sortie, la fonction `inv_interp_EF_ddl` pour donner les composantes d'une fonction d'un espace EF produit E à partir de ses valeurs aux *ddl* (utilisation dans le cas des espaces de type espaces plus fonctions «bulles») et réalise des tests d'inversion de la fonction `interp_EF_ddl` en utilisant la fonction la fonction `inv_interp_EF_ddl`. La commande pour cet exemple est

```
cd C :/EF/exemples/trace_graphe_interp, exemple2_trac_solEc ;
```

Exemple 4 : Variante de `exemple_interp_EF_ddl`. La commande pour cet exemple est

```
cd C :/EF/exemples/trace_graphe_interp, ...
    exemple_fonctions_bulles ;
```

Ce fichier effectue le tracé, avec la fonction `trac_solEc`, des fonctions de type bulles à partir de leurs composantes dans la base de l'espace EF produit E et donne des exemples d'utilisation de `interp_EF` et de sa fonction réciproque la fonction `inv_interp_EF_ddl`. Le fichier *script* de l'exemple 4 est le suivant.

EF/exemples/trace_graphe_interp/exemple_fonctions_bulles.m (exemple 4)

```
%Fichier EF/exemples/trace_graphe_interp/exemple_fonctions_bulles.m
%Exemple d'utilisation de INTERP_EF_DDL, INV_INTERP_EF_DDL et
%  TRAC_SOLEC sur des espaces avec des fonctions de type bulles.
%  On se place sur l'espace E = E1 x E2 x E3 x E4 x E5 x E6
%  donne par :
%
%      E1 = P1 + 1 bulles P3 continu = F1,
%      E2 = P1 + 3 bulles P4 continu = F2,
%      E3 = P1 + 3 bulles P4 continu = F2,
%      E4 = P1 + 3 bulles P4 continu = F2,
%      E5 = P1 + 1 bulles P3 discontinu = F3,
%      E6 = P2 continu = F4.
%
%  On considere en exemple la fonction f = (f1,f2,f3,f4,f5,f6)
%  de E, avec :
%
%  f1 : fonction qui prend la valeur de la fonction bulle P3 sur
%  chaque triangle,
%
%  f2 : fonction qui prend la valeur de la premiere fonction
%  bulle P4 sur chaque triangle,
%
%  f3 : fonction qui prend la valeur de la somme des deux
%  premieres fonctions bulles P4 sur chaque triangle,
%
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/interp_EF_ddl

```
% f4 : fonction somme de la fonction x + y et de la fonction
% qui prend la valeur de la somme des trois premieres fonctions
% bulles P4 sur chaque triangle et qui est alors egale a la
% fonction bulle P3. On a donc f3 = x + y + f1,
%
% f5 : fonction dont les composantes sur E5 sont nulles sauf
% celles des fonctions bulles P3, on a alors f5 = f1,
%
% f6 : fonction qui represente la somme des fonctions bulles du
% P2 chacune "a cheval" sur les aretes du maillage.
%
% La fonction INV_INTERP_EF_DDL permet d'obtenir les composantes
% de x + y pour le calcul des composantes de f4.
%
% La fonction TRAC_SOLEC effectue, sur 6 figures distinctes,
% le trace des 6 composantes la fonction f .
%
% On compose INTERP_EF_DDL avec INV_INTERP_EF_DDL et on verifie
% que l'on obtient l'identite.
```

```
clear ;
```

```
close all ;
```

```
%Choix des types d'espaces EF : F1, F2, F3 (bulles), F4 :
```

```
D.E = [13,14,113,2] ;
```

```
%Choix de l'espace produit E defini par
```

```
% E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
```

```
D.f = [1,2,2,2,3,4] ;
```

```
%Definition du maillage et des variables de maillage T, X, Y :
```

```
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
```

```
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
```

```
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
```

```

Noeu = length(X) ;

%Calcul du terme frontiere Front du maillage qui est ici facultatif
% pour le calcul des DDL qui suit :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;

%Calcul de la structure des DDL (on peut aussi poser Front = [] ci-
% dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Initialisation de diverses valeurs de Uc par ses composantes :
%-----%
%f1) P1 + bulle P3. Les composantes associees aux fonctions bulles
% du P3 valent 1 et les autres composantes valent 0 (les DDL du
% P1 sont numerotes de 1 a 9) :
Uc{1} = 0*Xc{1}' ;
Uc{1}(10 :end) = 1 ;

%f2) P1 + 3 bulle P4. Les composantes associees a la premiere
% fonctions bulles P4 de chaque triangle valent 1 et les autres
% composantes valent 0 :
Uc{2,1} = 0*Xc{2}' ;
Uc{2}(10 :3 :end) = 1 ;

%f3) P1 + 3 bulle P4. Les composantes associees aux deux premieres
% fonctions bulles P4 de chaque triangle valent 1 et les autres
% composantes valent 0 :
Uc{3,1} = 0*Xc{2}' ;

```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/interp_EF_ddl

```

Uc{3}(10 :3 :end) = 1 ;
Uc{3}(11 :3 :end) = 1 ;

%f4) P1 + 3 bulle P4. Les composantes associees aux fonctions
%   bulles du P4 valent 1 et les autres composantes asociees
%   au P1 sont celles de la fonction x + y. La fonction est
%   alors la fonction x + y a laquelle on ajoute une
%   fonction bulle du P3 sur chaque triangle :
Uc{4} = Xc{2}' + Yc{2}' ;
Ub = inv_interp_EF_ddl(donnees, Uc, 0, 4, X, Y, T, Xc, Yc, Tc) ;
Uc{4} = Ub{4} ;
Uc{4}(10 :end) = 1 ;

%f5) P1 + bulle P3 discontinu. Les composantes associees aux
%   fonctions bulles du P3 valent 1 et les autres composantes
%   valent 0 :
Uc{5} = 0*Xc{3}' ;
Uc{5}(Tc{3}( :,4)) = 1 ;

%f6) P2. Les composantes associees aux DDL d'aretes du P2 valent 1
%   et celles associees au sommets du P2 valent 0. On obtient alors
%   la somme des fonctions bulles d'aretes du P2 :
Uc{6} = 0*Xc{4}' ;
Uc{6}(10 :end) = 1 ;

%Traces des six fonctions obtenues :
trac_solEc(donnees, Uc, 0, 1 :6, 32, [], [], X, Y, T, Tc) ;

%Interpolation aux DDL :
U = interp_EF_ddl(donnees, Uc, 0, [], X, Y, T, Xc, Yc, Tc) ;

%Operation inverse : calcul des composantes a partir des valeurs
%   aux DDL :
Uc2 = inv_interp_EF_ddl(donnees, U, 0, [], X, Y, T, Xc, Yc, Tc) ;

```

```

%Uc{i} et Uc2{i} sont identiques par composition de interp_EF_ddl
% et de sa reciproque inv_interp_EF_ddl :
disp(['Ecart apres composition de interp_EF_ddl avec ' ...
      'inv_interp_EF_ddl : passage des composantes aux ', ...
      'valeurs aux DDL puis passage des valeurs aux composantes :']);
for i=1 :6,
    disp(['element numero : ', ...
          num2str(D.E(D.f(i))))], disp(max(abs(Uc{i} - Uc2{i})))));
end;
format;

```

Algorithme :

EF/trace_graphe_interp/interp_EF_ddl.m

On remet ici la partie déclarative de la fonction :

```

function [U, varargout] = interp_EF_ddl(donnees, Uc, varargin);
• Tests sur les variables et gestions des variables optionnelles, les variables der, r,
X, Y, T et Tc sont alors initialisées par chargement, calcul ou variables d'entrée.
%Initialisation de U a la taille de Uc
for i=1 :length(Uc)
    U{i,1} = [] ;
end;
• rep(i) ci-dessous est le numéro d'espace d'EF du i-ème espace d'EF  $E_i$  de
l'espace EF produit  $E$ .
D = donnees.D ;
rep = D.E(D.f) ;
• Calcul des variables cas, Deg1 et Deg2 suivantes : cas est la liste des numéros des
espaces d'EF programmés. L'espace numéro cas(k) est alors sur chaque triangle
K l'espace  $\mathcal{P}_{i,j}(K)$  du Paragraphe 1.4 avec  $i = \text{Deg1}(k)$ ,  $j = \text{Deg2}(k)$  :
U9 = 1 :9 ;
U10 = 0 :9 ;
Un10 = ones(1,10) ;
%Cas continu espaces 1 a 99
cas = [U9,10 :99] ;
Deg1 = [U9,Un10,2*Un10,3*Un10,4*Un10,5*Un10, ...

```

```

6*Un10,7*Un10,8*Un10,9*Un10] ;
Deg2 = [U9,U10,U10,U10,U10,U10,U10,U10,U10] ;
%ajout du cas discontinu : espaces numero 100 a 199
cas = [cas, 100, cas + 100] ;
Deg1 = [Deg1, 0, Deg1] ;
Deg2 = [Deg2, 0, Deg2] ;
• Calcul de la variable cas2 correspond aux EF créés par l'utilisateur, ici les
espaces d'EF  $P_{10}$ -continu et  $P_{10}$ -discontinu de numéros respectifs 200 et 300.
Pour la création de nouveaux éléments finis dans le code on pourra aussi utiliser
la fonction interp_EF_ddl2 qui ne nécessite pas de modifications pour ce type
d'ajout. On pose alors :
cas2 = [200,300] ;
• Boucle en  $k$  sur les triangles :
for k=1 :size(T,1)%Nombre de triangles
- Calcul des coordonnées  $(a_1, a_2)$ ,  $(b_1, b_2)$ ,  $(c_1, c_2)$  des sommets du triangle
courant numéro  $k$  (pour le maillage EF) et de la variable  $\mathbf{de} = (a_1, b_1, c_1) \cdot (b_2 - c_2, -a_2 + c_2, -b_2 + a_2)$  (le symbole  $\cdot$  désigne ici le produit scalaire) à partir des
coordonnées des sommets du triangle courant donnés par M1 et M2 ci-dessous :
M1 = X(T(k,1 :3)) ;
M2 = Y(T(k,1 :3)) ;
a1 = M1(1) ;
b1 = M1(2) ;
c1 = M1(3) ;
a2 = M2(1) ;
b2 = M2(2) ;
c2 = M2(3) ;
de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;
• Boucle sur les composantes de l'espace EF produit :
for i=1 :length(rep)
if any(i==r)%Si i appartient a r ou vec faire :
TT = Tc{D.f(i)}(k, :) ;
V = Uc{i} ;
d = der ; %derivation
- Détermination en fonction de rep(i) des termes deg1 et deg2 qui définissent
l'espace  $\mathcal{P}_{deg1,deg2}(K)$  du Paragraphe 1.4.
e = rep(i) ; %e : numero de l'epace EF courant

```



```

I = find(e==cas) ; %si non vide : famille 1
I2 = find(e==cas2) ; %si non vide : famille 2
- Calcul des valeurs des coordonnées barycentriques des ddl de l'espace d'EF
courant :

```

```

if ~isempty(I)%Si famille 1 :
    deg1 = Deg1(I) ; %Les fonctions de bases
                    %sont definies par :
    deg2 = Deg2(I) ; %degres : "polynomes d'aretes"
                    %et "bulles"
%Calcul des coordonnees barycentriques des ddl d'aretes
L1b = [1, 0, 0, (deg1-1 :-1 :1)/deg1, ...
        (deg1 -1 :-1 :1)/deg1, zeros(1,deg1-1)] ;
L2b = [0, 1, 0, (1 :deg1-1)/deg1, ...
        zeros(1,deg1-1), (deg1-1 :-1 :1)/deg1] ;
L3b = 1 - L1b - L2b ;
%Parcours des DDL interieurs et calcul des
%coordonnees barycentriques de ces DDL :
compt = 0 ;
z1 = [] ;
z2 = [] ;
z3 = [] ;
for k1=1 :deg2 - 2,
    for i1=deg2 - k1 - 1 :-1 :1,
        j1=deg2 - k1 - i1 ;
        compt = compt+1 ;
        z1(compt) = i1/deg2 ;
        z2(compt) = j1/deg2 ;
        z3(compt) = k1/deg2 ;
    end ;
end ;
L1b = [L1b,z1] ;
L2b = [L2b,z2] ;
L3b = [L3b,z3] ;

```

- Valeur de la composante numéro i de la fonction aux *ddl* de l'espace d'EF
numéro $\text{rep}(i)$ à partir de la donnée des coordonnées barycentriques de ces points,

4. DESCRIPTION DU CODE FONCTION `trace_graphe_interp/interp_EF_ddl2`

du numéro de l'espace d'EF (variable `e`) et des paramètres du triangle courant (triangle numéro `k`) :

```
z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, ...
          L1b, L2b, L3b, deg1, deg2) ;
```

```
%Calcul de U{i}
```

```
U{i,1}(TT) = sum(diag(V(TT))*z,1)';
```

- Cas de la deuxième famille d'espaces d'EF :

```
elseif ~isempty(I2)%Si famille 2
```

```
    deg1 = 10 ;
```

```
    deg2 = 10 ;
```

On réitère le même processus que pour la famille 1 avec ces valeurs de `deg1` et `deg2`.

```
end%Fin du si (if ~isempty(I))
```

```
end%Fin du si (if any(i==r))
```

```
end%Fin de la boucle en j sur les composantes d'espace
```

```
end%Fin de la boucle en k sur les triangles
```

- Transposition du vecteur ligne contenu dans chaque cellule `U` :

```
for i=1 :length(U)
```

```
    U{i} = U{i}' ;
```

```
end ;
```

et gestion des variables de sortie optionnelles `Xc`, `Yc` et `Tc`.

Fonction :

```
function [U, varargout] = interp_EF_ddl2(donnees, Uc, varargin) ;
```

Cette fonction est identique à la fonction `interp_EF_ddl` à l'erreur machine près, seul l'algorithme est modifié comme indiqué aux points 1) et 2) de la description de l'algorithme qui suit.

Par rapport à `interp_EF_ddl`, il n'est pas nécessaire de modifier `interp_EF_ddl2` pour la prise en compte de l'ajout de nouveaux types d'espaces d'EF.

Algorithme :

`EF/trace_graphe_interp/interp_EF_ddl2.m`

On remet ici la partie déclarative de la fonction :

```
function [U, varargout] = interp_EF_ddl2(donnees, Uc, varargin) ;
```

- Tests sur les variables et gestions des variables optionnelles, les variables `der`, `r`, `X`, `Y`, `T` et `Tc` sont alors initialisées par chargement, calcul ou variables d'entrée.

```
%Initialisation de U a la taille de Uc
for i=1 :length(Uc)
    U{i,1} = [] ;
end ;
```

- **rep(i)** ci-dessous est le numéro d'espace d'EF du *i*-ème espace d'EF E_i de l'espace EF produit E .


```
D = donnees.D ;
rep = D.E(D.f) ;
```
- Boucle en k sur les triangles :


```
for k=1 :size(T,1)%Nombre de triangles
    - Calcul des coordonnées  $(a_1, a_2)$ ,  $(b_1, b_2)$ ,  $(c_1, c_2)$  des sommets du triangle
      courant numéro  $k$  (pour le maillage EF) et de la variable  $\mathbf{de} = (a_1, b_1, c_1) \cdot (b_2 - c_2, -a_2 + c_2, -b_2 + a_2)$  (le symbole  $\cdot$  désigne ici le produit scalaire) à partir des
      coordonnées des sommets du triangle courant donnés par M1 et M2 ci-dessous :
      

```
M1 = X(T(k,1 :3)) ;
M2 = Y(T(k,1 :3)) ;
a1 = M1(1) ;
b1 = M1(2) ;
c1 = M1(3) ;
a2 = M2(1) ;
b2 = M2(2) ;
c2 = M2(3) ;
de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;
```


```
- Boucle sur les composantes de l'espace EF produit :


```
for i=1 :length(rep)
    if any(i==r)%Si i appartient a r ou vec faire :
        TT = Tc{D.f(i)}(k, :) ;
        V = Uc{i} ;
        d = der ;
        e = rep(i) ;
    - Calcul des valeurs des coordonnées barycentriques des ddl de l'espace d'EF
      courant du Paragraphe 1.4.
```

-
- 1) Par rapport à `interp_EF_ddl`, on calcule les coordonnées barycentriques des *ddl* par résolution du système que vérifient les coordonnées barycentriques :

$$X1 = Xc\{D.f(i)\}(TT) ;$$

```

Y1 = Yc{D.f(i)}(TT) ;
AMAT = [a1, b1, c1 ; a2, b2, c2 ; 1, 1, 1] ;
BMAT = [X1 ; Y1 ; ones(1,length(X1))] ;
CMAT = AMAT\BMAT ;
L1b = CMAT(1, :) ;
L2b = CMAT(2, :) ;
L3b = CMAT(3, :) ;

```

- Calcul de la valeur de la composante numéro i de la fonction aux *ddl* de l'espace d'EF numéro `rep(i)` à partir des coordonnées barycentriques de ces points, du numéro d'espace et des paramètres du triangle courant :

• 2) Par rapport à `interp_EF_ddl`, on utilise la fonction `F_base_T` à la place de la fonction `F_base`. Les points 1) et 2) de cet algorithme permettent de simplifier la prise en compte d'espaces d'EF supplémentaire dans cette fonction.

```

[z, dim] = F_base_T(e, d, [], [], a1, a2, b1, b2, ...
                    c1, c2, de, D, L1b, L2b, L3b) ;

```

```

U{i,1}(TT) = sum(diag(V(TT))*z,1)' ;
end%Fin du si (if any(i==r))
end%Fin de la boucle en j sur les composantes d'espace
end%Fin de la boucle en k sur les triangles

```

• Transposition du vecteur ligne contenu dans chaque cellule `U` et gestion des variables de sortie optionnelles `Xc`, `Yc`, `Tc`.

Ainsi, si l'on ajoute une espace d'EF dans le code, en prenant en compte de cet ajout dans `F_base_T`, on n'aura pas à modifier la fonction `interp_EF_ddl2`.

Fonction :

```
function [U, varargout] = inv_interp_EF_ddl(donnees, Uc, varargin) ;
```

Interpolation d'une fonction par une fonction de l'espace EF produit E . Fonction réciproque de `interp_EF_ddl`. Pour une fonction d'un espace EF, calcule, dans la base choisie dans le code EF, les composantes de cette fonction à partir de ses valeurs aux *ddl* de l'espace EF considéré. Pour une fonction quelconque définie aux *ddl* de l'espace EF E on obtient à partir de ses valeurs aux *ddl* son interpolé dans l'espace E .

Soit $E = E_1 \times \dots \times E_i \times \dots \times E_n$ l'espace EF produit décrit par la variable `donnees` et soit f une fonction continue de $\mathbb{R}^2$ à valeurs dans $\mathbb{R}^n$ dont les valeurs aux *ddl* de E sont décrites par la variable `Uc`, alors `inv_interp_EF_ddl` permet de calculer les composantes de la fonction de E qui interpole f aux *ddl* de E en inversant le problème d'interpolation qui en résulte.

Les commandes sont identiques à celles de la fonction `interp_EF_ddl` seuls les rôles de `U` et `Uc` permutent. Ainsi la commande :

```
U = inv_interp_EF_ddl(donnees, Uc) ;
```

renvoie le vecteur de type `CELL` `U` tel que pour chaque valeur de i , `U{i}` est le vecteur des composantes dans la base de E_i le i -ème espace d'EF de l'espace EF produit E , obtenu à partir des valeurs aux *ddl* de E_i données par le vecteur `Uc{i}`.

La commande longue est donnée par

```
inv_interp_EF_ddl(donnee, Uc, der, r, X, Y, T, Xc, Yc, Tc) ;
```

Nous avons reporté ci-dessous les changements par rapport à `interp_EF_ddl` qui portent sur les variables `Uc` et `U` dont les définitions permutent, ainsi que les modifications dans l'algorithme.

La dérivation, indiquée par la variable `der` dans la fonction `interp_EF_ddl`, est ici purement indicative : le problème d'interpolation posé dans l'algorithme est alors celui de chercher un élément de E dont la dérivée réalise l'interpolation qui conduit à des matrices `A{i}` des systèmes obtenus et décrite dans l'algorithme ci-dessous qui ne sont pas inversibles.

Pour des exemples d'utilisation de la fonction `inv_interp_EF_ddl`, on se reportera à ceux donnés dans la description de la fonction `interp_EF_ddl` qui utilisent aussi la fonction `inv_interp_EF_ddl`.

Variables d'entrée :

`Uc` : `CELL ARRAY`. Vecteur colonne. Le terme `Uc{i}` est le vecteur colonne de type `DOUBLE ARRAY` des valeurs aux *ddl* du i -ème espace d'EF de l'espace EF produit E , de la i -ème composante d'une fonction. Le vecteur `Uc` peut aussi avoir des cellules vides.

Variables de sortie :

U : `CELL ARRAY`. Vecteur colonne de même taille que **Uc**. Lorsqu'il est calculé, le terme $U\{i\}$ est le vecteur colonne de type `DOUBLE ARRAY` des composantes, dans la base du i -ème espace d'EF de l'espace EF produit E , de la fonction dont la valeur aux *ddl* de E est donnée par $U\{i\}$. Sinon $U\{i\} = []$.

Si la variable optionnelle **r** est vide ou non utilisée alors les cellules de **Uc** doivent être toutes initialisées et toutes les cellules de **U** sont alors calculées.

Sinon, les cellules de **Uc** dont le numéro appartient à **r** doivent au moins être initialisées **Uc** et ce sont les cellules correspondantes de **U** qui sont calculées.

Exemples. On pourra se reporter aux trois exemples donnés à la description de la fonction `interp_EF_ddl`.

L'exemple ci-dessous reprend l'espace EF produit E de ces exemples :

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

avec le maillage de la figure 4.3-5. Il effectue le calcul de l'interpolée sur E de la fonction

$$g = (x + y, -y, x, e^{xy}, x^4 + y^4),$$

notée f . Dans cet exemple les composantes de f sont calculées par la commande raccourcie :

```
Uc = inv_interp_EF_ddl(donnees, Ug) ;
```

avec **Ug** valeurs de g aux *ddl* de E . Ensuite les valeurs de f aux *ddl* de E sont calculées par la commande raccourcie :

```
Uf = interp_EF_ddl(donnees, Uc) ;
```

On vérifie alors que $Ug = Uf$. Le fichier est le suivant :

EF/exemples/trace_graphe_interp/exemple_inv_interp_EF_ddl.m

```
%Fichier EF/exemples/trace_graphe_interp/exemple_interp_EF_ddl.m
```

```
%Exemple d'interpolation avec INV_INTERP_EF_DDL et de calcul
```

```
% de valeurs aux DDL avec INTERP_EF_DDL sur l'espace d'EF
```

```
% E = E1 x E2 x E3 x E4 x E5, avec :
```

```
%
```

```
% E1 = P1 continu = F1,
```

```
% E2 = P2 continu = F2,
```

```
% E3 = P2 continu = F2,
```

```
% E4 = P1 + 3 bulles P4 continu = F3,
```

```
% E5 = P3 discontinu = F4.
```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/inv_interp_EF_ddl

```
%
% Les calculs sont effectues sur la fonction g definie par :
%
%           $g = (x + y, -y, x, \exp(xy), x^4 + y^4).$ 
%
% - La fonction INV_INTERP_EF_DDL calcule les composantes  $U_c$  de
% la fonction  $f = I_E(g)$  element de  $E$  qui vaut  $g$  aux DDL de  $E$ .
%
% - On verifie avec INTERP_EF_DDL applique a  $U_c$  que  $g$  et  $f$ 
% coincident aux DDL de  $E$ .
%
% On utilise ici des champs donnees.nom_prep et donnees.nom_mail
% pour utiliser les commandes raccourcies de INTERP_EF_DDL et de
% INV_INTERP_EF_DDL.

clear ;
close all ;

%Choix des types d'espaces EF F1, F2, F3 (bulles) et F4 :
D.E = [1,2,14,103] ;

%Choix  $E = E_1 \times E_2 \times E_3 \times E_4 \times E_5 = F_1 \times F_2 \times F_2 \times F_3 \times F_4$  :
D.f = [1,2,2,3,4] ;

%Definition du maillage et de ses variables T, X, Y, Noeu Front :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;
Front = {[7,8,9] ; [9,6,3] ; [3,2,1] ; [1,4,7]} ;

%Choix des formules d'integration, derivation (pour EF_Prep) :
D.intT = 10 ;
D.intA = 6 ;
```

4. DESCRIPTION DU CODE FONCTION trace_graphe_interp/inv_interp_EF_ddl2

```

D.der = [0,1,2] ;

%Initialisation des champs de donnees necessaires a EF_Prep :
donnees.D = D ;
donnees.nom_mail = ...
    'C :/EF/trace_graphe_interp/fich_trac_solEc/f_mail_inv' ;
donnees.nom_prep = ...
    'C :/EF/trace_graphe_interp/fich_trac_solEc/f_prep_inv' ;

%Creation du fichier de maillage
save(donnees.nom_mail,'T','X','Y','Front','Noeu')

%Calcul du fichier relatif aux espaces d'EF et aux formules
% d'integration :
EF_Prep(donnees) ;

%Chargement des coordonnees des DDL :
load(donnees.nom_prep,'Xc','Yc') ;

%Calcul de Uc representant la fonction  $f = I_E(g)$  :
%-----%
Ug{1} = Xc{1}' + Yc{1}' ;
Ug{2} = -Yc{2}' ;
Ug{3} = Xc{2}' ;
Ug{4} = exp(Xc{3}'.*Yc{3}') ;
Ug{5} = Xc{4}'.^4 + Yc{4}'.^4 ;
Ug = Ug' ;
Uc = inv_interp_EF_ddl(donnees, Ug) ;

%Valeurs aux DDL de E de la fonction f de E :
%-----%
Uf = interp_EF_ddl(donnees, Uc) ;
disp(['On verifie que g et son interpolee coincident aux DDL :']);
for i=1 :5,
    disp(['element numero : ', num2str(D.E(D.f(i)))]);
    disp(max(abs(Ug{i} - Uf{i})));
end ;

```


4.3. DESCRIPTION DES FONCTIONS FN `trace_graphe_interp/inv_interp_EF_ddl2`

Fonction :

```
function [U, varargout] = inv_interp_EF_ddl2(donnees, Uc, ...  
                                             varargin) ;
```

Cette fonction est identique à `inv_interp_EF_ddl`, seul l'algorithme est modifié de la même manière que dans `interp_EF_ddl`, voir cette fonction.

Fonction :

```
function [Vc, varargout] = proj_deriv_EF(donnees, Uc, der, r, ...
                                         nfig, varargin);
```

Dérivation par projections orthogonales L^2 successives.

La commande

```
Vc = proj_deriv_EF(donnees, Uc, der, r, nfig);
```

projette sur l'espace EF produit E défini par la variable `donnees`, les dérivées des composantes `r(1)`, `r(2)`, *etc.* de la fonction représentée par la variable `Uc` relativement à la base choisie dans le code EF pour E .

La variable `der` est vecteur de cellules. Si i n'est pas une composante de `r` ou si `der{i}` = [], alors

$$\mathbf{Vc}\{i\} = \mathbf{Uc}\{i\}.$$

Pour `r` = [] ce sont toutes les composantes de `Uc` qui sont prises en compte : `r` = [] est équivalent à `r` = 1 : `length(Uc)`.

Ensuite on a deux cas.

Cas 1. Le terme `der{i}` est un scalaire à valeur dans $\{0, 1, 2\}$. Soit i une des composantes de `r` et E_i le i -ème espace de l'espace EF produit E . Soit V la fonction de E_i de composantes `Uc{i}` dans la base de E_i . Alors `Vc{i}` est la composante dans la base de E_i de la solution V_e de :

$\exists V_e \in E_i$ tel que

$$\int_{\Omega} V_e W = \int_{\Omega} V_{,der\{i\}} W, \quad \forall W \in E_i.$$

La fonction V_e est alors égale à V si `der{i}` = 0 ou [] et c'est une approximation de $V_{,x}$ si `der{i}` = 1 et de $V_{,y}$ si `der{i}` = 2. Pour `der{i}` = 0 la projection est effectivement calculée et `Vc{i}` peut différer de `Uc{i}` à la précision machine près.

Cas 2. Le terme `der{i}` est un vecteur dont les composantes sont à valeurs dans $\{0, 1, 2\}$.

Dans ce cas, V_e sera alors solution des projections successives :

$$V_e^0 = V,$$

Pour $k = 1, \dots, d = \text{der}\{i\}(\text{end})$, V_e^k est la solution de

$$\exists V_e^k \in E_i, \int_{\Omega} V_e^k W = \int_{\Omega} V_{e,der\{i\}(k)}^{k-1} W, \quad \forall W \in E_i,$$

$$V_e = V^d.$$

La fonction V_e représente alors la dérivation $V_{,der\{i\}(1)} \dots der\{i\}(\text{end})$ (cette notation est formelle car V n'est pas dérivable sauf dans des cas particuliers).

Pour `r` = [] ce sont toutes les composantes de `Uc` qui sont dérivées.

La variable `nfig` donne des indications sur le tracé et les figures. Si `nfig` est un entier non nul, les figures sont tracées avec une numérotation commençant par `nfig` (1 si `nfig` = []). Si `nfig` est de type `struct` alors les figures sont toutes tracées sur la figure numéro `nfig.n` qui comporte un tableau de `nfig.t(1) × nfig.t(2)` figures. Si `nfig` = 'n' ou `nfig.n` = 0, il n'y a pas de tracé. Les tracés se font par subdivision des triangles du maillage en 16 sous-triangles (subdivision fixe). Pour d'autres sous-maillages, on pourra utiliser la fonction `trac_solEc` avec comme variable d'entrée représentant les composantes de la fonction à tracer la variable `Vc` calculée par `proj_deriv_EF`.

La variable `donnees` est la variable qui décrit le problème d'approximation EF. La commande ci-dessus utilise le fichier relatif aux espaces d'EF et aux formules de quadrature indiqué par le champ `donnees.nom_prep`. Le calcul de la projection utilise la variable `SL_Dom` de ce fichier et le tracé, les variables de `ddl` `Xc`, `Yc` et `Tc`. La variable `SL_Dom` représente les matrices d'assemblage pour la projection.

La commande ci-dessus utilise aussi, pour le tracé, le fichier de maillage de nom `donnees.nom_mail`. Pour `nfig` = 'n', le tracé n'est pas effectué, le champ `donnees.nom_mail`, le fichier correspondant et les variables de `ddl` `Xc`, `Yc` et `Tc` sont alors facultatifs.

La commande

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig);
```

permet de plus de récupérer les matrices d'assemblage en sortie. La variable de sortie `SL_Dom` est une variable optionnelle.

La commande suivant permet de calculer V_e à partir de la donnée de la variable `SL_Dom` mise en entrée.

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, SL_Dom);
```

Les champs `donnees.nom_prep` et par `donnees.nom_mail` ainsi que les fichiers désignés par ces fichiers ne sont alors nécessaires qu'en cas de tracé.

Dans les commandes qui suivent, il n'est pas nécessaire d'initialiser les champs `donnees.nom_prep` et `donnees.nom_mail` ainsi que de créer les fichiers qu'ils désignent.

Les deux commandes suivantes permettent de calculer V_e à partir de `SL_Dom` :

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             SL_Dom, X, Y, T);
```

ou

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             SL_Dom, X, Y, T, Xc, Yc, Tc);
```

Les calculs se font à partir des variables de maillage `X`, `Y` et `T`. En cas de tracé, la seconde commande est plus rapide que la précédente car les `ddl` ne sont alors pas recalculés. La variable `SL_Dom` est utilisée pour la projection, elle n'est pas

recalculée comme dans les deux commandes qui suivent où on pose `SL_Dom = []` en variable d'entrée.

Les deux commandes suivantes permettent de recalculer la variable `SL_Dom` à partir des variables du maillage `X, Y, T` :

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             [], X, Y, T) ;
```

ou

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             [], X, Y, T, Xc, Yc, Tc) ;
```

Cette commande est plus rapide que la précédente car les *ddl* ne sont pas recalculés.

Remarque. Les *ddl* peuvent s'obtenir par

```
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []);
```

Variables d'entrée :

`donnees` : STRUCT ARRAY. Variable décrivant les données du problème EF. Les champs à initialiser sont les champs décrivant le choix de l'espace EF produit *E* :

```
donnees.D.E,
donnees.D.f,
```

et ensuite suivant l'appel, on initialise les champs suivants.

Pour un appel, sans les variables de maillage et les variables *ddl* du type

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig) ;
```

les champs à initialiser sont

```
donnees.nom_prep,
```

et pour le tracé

```
donnees.nom_mail.
```

Le fichier désigné par `donnees.nom_prep` doit contenir la variable `SL_Dom` et, si on veut effectuer un tracé, la variable `Tc`, le tableau décrivant les *ddl*. Pour le tracé, le fichier de maillage désigné par `donnees.nom_mail` doit contenir les variables de maillage `X, Y, T`.

Pour un appel du type

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, SL_Dom) ;
```

avec `SL_Dom` variable contenant les matrices d'assemblage pour la projection, les champs à initialiser sont :

```
donnees.nom_mail,
donnees.nom_prep,
```

et ce, uniquement pour tracer.

Pour un appel avec les variables de maillage du type

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, [], ...
                             X, Y, T);
```

ou

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, [], ...
                             X, Y, T, Xc, Yc, Tc);
```

les champs à initialiser sont :

```
donnees.D.der,
donnees.D.intT.
```

pour le calcul de SL_Dom.

Pour un appel avec les variables de maillage du type

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             SL_Dom, X, Y, T);
```

ou avec en plus les variables décrivant les *ddl*

```
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, SL_Dom, ...
                             X, Y, T, Xc, Yc, Tc);
```

il n'y a pas d'autres champs à initialiser que `donnees.D.E` et `donnees.D.f`, tracés compris.

Uc : CELL ARRAY. Vecteur colonne dont les cellules sont des vecteurs colonnes de type DOUBLE ARRAY. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (cf. Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

der : CELL ARRAY. `der{i}` est un vecteur ligne ou colonne dont les termes valent [], 0, 1 ou 2 indiquant la dérivation. [] est mis pour 0. Les valeurs 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

r : DOUBLE ARRAY. Variable portant sur les composantes de l'espace EF produit E . Vecteur ligne ou colonne, `r(i)` désigne la composante de la fonction à projeter et à tracer. Si i n'est pas une composante de **r** avec **r** non vide alors la variable de sortie **Vc** vérifie `Vc{i} = Uc{i}`. Pour `r = []` toutes les composantes de **Uc** sont prises en compte.

nfig : DOUBLE ARRAY ou STRUCT ARRAY. Variable de figure. Lorsque **nfig** est de type DOUBLE ARRAY, la composante `r(i)` de la fonction à tracer est tracée sur la figure numéro i si `nfig = []` et sur la figure numéro `nfig-1+i` si **nfig** est un entier. Lorsque **nfig** est de type STRUCT ARRAY, `nfig.n` indique le numéro de figure sur

lesquelles seront tracés tous les graphes dans un tableau de `nfig.t(1) × nfig.t(2)` figures. Si `nfig = 'n'` ou `nfig.n = 0`, il n'y a pas de tracé. Les tracés se font par subdivision des triangles du maillage en 16 sous-triangles.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme : `SL_Dom` : `CELL ARRAY`. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par `donnees.D.E(k)`. Le terme `SL_Dom{i,j,k,m}` est une matrice de type `SPARSE ARRAY` de dimension $\dim(F_i) \times \dim(F_j)$. La matrice `SL_Dom{i,j,k,m}` est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par

$$\int_{\Omega} v_{,k-1} w_{,m-1}.$$

Le terme (p, q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$, où $\{u^k\}_k$ désigne la base de l'espace d'EF F_k . Les bases sont décrites au Paragraphe 1.4 et au Paragraphe 2.3.2, sur un maillage exemple. Cette variable représente le champ `SL_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

`X,Y` : `DOUBLE ARRAY`. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`Xc, Yc` : `CELL ARRAY`. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type `DOUBLE ARRAY`. Les vecteurs `Xc{i}` et `Yc{i}` sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

`Tc` : `CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

`Vc` : `CELL ARRAY`. Vecteur colonne. Si i est une composante de $\mathbf{r}$ ou si $\mathbf{r} = []$ alors le terme `Vc{i}` est le vecteur colonne des composantes dans E_i , le i -ème espace de l'espace EF produit E , de la représentation des dérivées d'ordre `der{i}` de la i -ème composante de la fonction représentée par `Uc`. Si i n'est pas une composante de $\mathbf{r}$ avec $\mathbf{r}$ non vide alors la variable de sortie `Vc` vérifie `Vc{i} = Uc{i}`.

Variables de sortie optionnelles :

La variable de sortie optionnelle prend la valeur de `SL_Dom` ci-dessus.

Exemple 1 : Cet exemple est tiré du `help` de la fonction. Cet exemple trace les dérivées d'ordre 0 et par rapport à x et y obtenues avec `proj_deriv_EF` de la

solution du système à deux inconnues calculé par l'exemple donné par le fichier *exemple_systeme_Bord_cell.m*. Rappelons le système (découplé) sur le domaine $\Omega =]0, 1[^2$ de frontière notée Γ :

$$-\Delta u_1 = 2x(1-x) + 2y(1-y), \quad u_1 = 0 \text{ sur } \Gamma,$$

$$-\Delta u_2 = 2x(1-x) + 2y(1-y), \quad u_2 = 0 \text{ sur } \Gamma,$$

de solution

$$u_1(x, y) = u_2(x, y) = x(1-x)y(1-y).$$

Le domaine Ω est muni du maillage du carré unité par 8 triangles de la figure 4.3-5 et on a posé comme espace EF produit E pour l'approximation du problème : $E = E_1 \times E_2$, où l'on a pris E_1 et E_2 égaux à $F_1 = P_4^c$, le P_4 continu. Cet exemple calcule et donne le tracé des projections de u_1 , u_2 , $u_{1,x}$ et $u_{2,y}$. L'exemple du `help` de la fonction `proj_deriv_EF` est alors le suivant :

EXEMPLE 1

```
%Traitement d'une solution EF : celle calculee par
% le fichier exemple_systeme_Bord_cell.
% On utilise le fichier de nom donnees.nom_prep calcule par
% exemple_systeme_Bord_cell pour recuperer la matrice
% EF SL_Dom.
% On utilise le fichier de nom donnees.nom_mail calcule par
% exemple_systeme_Bord_cell pour recuperer le maillage et
% effectuer les traces.
exemple_systeme_Bord_cell;
close all;
disp('Trace de solution du probleme (figures numeros 1 et 2)');
der{1} = [0];
der{2} = [0];
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, [], 1);
disp('Derivee par rapport a : x de u1, y de u2 (fig. 3 et 4)');
der{1} = [1];
der{2} = [2];
[Vc2,SL_Dom2] = proj_deriv_EF(donnees, Uc, der, [], 3);
```

Exemple 2 : Cet exemple est mis en œuvre par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple2_proj_deriv_EF;
```

et reprend la représentation de la fonction de u de l'exemple 1 ci-dessus sans utiliser le fichier *script*.

EF/exemples/Assemblage/exemple_systeme_Bord_cell.m.

Pour cela on initialise la variable `donnees.D`, les variables de maillage, on calcule les variables qui donnent les *ddl* et on initialise `Uc` aux composantes dans la base EF de la fonction de E :

$$u_1(x, y) = u_2(x, y) = x(1 - x)y(1 - y).$$

Ensuite, on sauvegarde les variables de maillage dans le fichier de maillage dont le nom est donné par `donnees.nom_mail` puis, avec la fonction `proj_deriv_EF`, on calcule la variable `SL_Dom` qui est sauvegardée avec la variable de *ddl* `Tc` dans le fichier de données relatif aux espaces d'EF et aux formules de quadrature dont le nom est donné par `donnees.nom_prep`. Comme ci-dessus, cet exemple donne le tracé des projections de u_1 , u_2 , $u_{1,x}$ et $u_{2,y}$, ainsi aussi que celui des fonctions $u_{1,xyxy}$, $u_{2,yyx}$ qui valent 4, à resp. $1.3 - 10^{-11}$ et $8.6 - 10^{-11}$ près en norme du max aux *ddl*. Le fichier exemple est le suivant.

EF/exemples/trace_graphe_interp/exemple2_proj_deriv_EF.m

```
%Fichier
%   EF/exemples/trace_graphe_interp/exemple2_proj_deriv_EF.m
%   Utilisation de PROJ_DERIV_EF : calcul et traces de projections
%   L2 de fonctions dans l'espace EF et de leur derivees.
%   La projection se fait sur l'espace E = E1 x E2 avec
%   E1 = E2 = P4 continu = F1 et la fonction projetee est :
%       u = (xy(1 - x)(1 - y), xy(1 - x)(1 - y)).

clear ;
close all ;

%Choix du types d'espaces, F1 = P4 continu :
donnees.D.E = [4] ;

%Choix de l'espace produit E = E1 x E2 = F1 x F1 = P4 x P4 :
donnees.D.f = [1,1] ;

%Choix de la formule d'integration :
donnees.D.intT = 10 ;
```



```

%Choix des derivations :
donnees.D.der = [0,1,2] ;

%Construction du maillage : triangles et coordonnees T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;

%Calcul des DDL :
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, X, Y, T, []);

%Initialisation de la fonction a derivier :
Uc{1} = Xc{1}'.*(1 - Xc{1}')*.Yc{1}'.*(1 - Yc{1}') ;
Uc{2,1} = Xc{1}'.*(1 - Xc{1}')*.Yc{1}'.*(1 - Yc{1}') ;

%Noms du fichier de maillage et du fichier relatif aux espaces
% EF et aux formules d'integration :
donnees.nom_mail = ['C :/EF/exemples/trace_graphe_interp', ...
                    '/fich_exemple_proj_deriv_EF/f_mail'] ;
donnees.nom_prep = ['C :/EF/exemples/trace_graphe_interp', ...
                    '/fich_exemple_proj_deriv_EF/f_prep'] ;

%Creation du fichier de maillage :
save(donnees.nom_mail, 'T', 'X', 'Y') ;

%Calcul de la variable SL_Dom :
der{1} = [0] ;
der{2} = [0] ;
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, [], 'n', [], ...
                             X, Y, T, Xc, Yc, Tc) ;

%Creation du fichier relatif aux espaces EF et aux formules
% d'integration, on ne stokes que les variables utilisees

```

```
% par PROJ_DERIV_EF :
save(donnees.nom_prep, 'SL_Dom', 'Tc');

%Traces :
disp('Trace de solution du probleme (2 figures numeros 1 et 2)');
der{1} = 0;
der{2} = 0;
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, [], 1);

disp('Derivees u1,x et u2,y (2 figures numeros 3 et 4)');
der{1} = 1;
der{2} = 2;
[Vc2,SL_Dom2] = proj_deriv_EF(donnees, Uc, der, [], 3);

disp('Derivees u1,xyy et u2,yyxx (2 figures numeros 5 et 6)');
der{1} = [1,1,2,2];
der{2} = [2,2,1,1];
[Vc3,SL_Dom3] = proj_deriv_EF(donnees, Uc, der, [], 5);

%Calcul d'erreur sur la derivation :
disp('max de |u1,xyy - 4| et de |u2,yyxx - 4|');
disp(max(abs(Vc3{1} - 4)));
disp(max(abs(Vc3{2} - 4)));
```

Exemple 3 : Cet exemple est donné par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple_proj_deriv_EF ;
```

Ce fichier donne un exemple calcul d'utilisation de la fonction `proj_deriv_EF` avec la variable de `SL_Dom` donnée en entrée.

Ce fichier reprend l'exemple 2 donné dans la description de la fonction `trac_solEc` où l'espace EF produit E est donné par le maillage du carré unité par 8 triangles de la figure 4.3-5 et

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

avec $P_{1,4}^c$ l'espace d'EF $P_1 \oplus 3$ bulles P_4 . et on considère la fonction $f = I_E(g)$ où g est la fonction

$$g = (x + y, -y, x, e^{xy}, x^4 + y^4),$$

et $I_E(g)$ la fonction de E qui interpole g aux *ddl* de E .

On calcule et on trace les composantes de la projection $[L^2(\Omega)]^5$ de la fonction $f = (f_1, f_2, f_3, f_4, f_5) \in E$, qui vaut aussi :

$$\begin{aligned} f_1(x, y) &= x + y, \quad f_2(x, y) = -y, \quad f_3(x, y) = x, \quad f_4(x, y) = I_{1,4}(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4), \end{aligned}$$

et de leur dérivées (au sens de la projection décrite ci-dessus dans le descriptif de la fonction `proj_der_EF`)

$$f_{i,x}, \quad f_{i,y}, \quad f_{i,xy} \quad \text{avec } 1 \leq i \leq 5,$$

où pour une fonction h continue $I_{1,4}(h)$ est la fonction qui interpole h aux *ddl* du $P_{1,4}^c$ et pour une fonction h continue sur les triangles du maillage $I_3(h)$ est la fonction du P_3^d qui interpole h aux *ddl* du P_3^d . Dans cet exemple pour calculer les composantes de f dans la base EF de l'espace E , on utilise la fonction `inv_interp_EF_ddl` qui résout le problème linéaire d'interpolation à partir des valeurs de g aux *ddl* de E .

EF/exemples/trace_graphe_interp/exemple_proj_deriv_EF.m

```
%Fichier  EF/trace_graphe_interp/exemple_proj_deriv_EF.m
%   Utilisation de PROJ_DERIV_EF : traces de projections L2 sur
%   un espace EF E de fonctions de cet espace et de leur derivees
%   avec E = E1 x E2 x E3 x E4 x E5 :
%
%       E1 = P1 continu = F1,
%       E2 = P2 continu = F2,
%       E3 = P2 continu = F2,
%       E4 = P1 + 3 bulles P4 continu = F3,
%       E5 = P3 discontinu = F4.
%-----%
%   Les calculs sont effectues sur l'interpolee sur E de la
%   fonction g definie par :
%
%       g = (x + y, -y, x, exp(xy), x^4 + y^4).
%
%   - La fonction INV_INTERP_EF_DDL calcule les composantes Uc de
%   la fonction f = I_E(g) element de E qui vaut g aux DDL de E.
%
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/proj_deriv_EF

```
% Calcul et trace avec PROJ_DERIV_EF de f et de representation
% par projection de f,x, f,y, f,xy

clear;
close all;
warning off;
disp(['Projection P1 de x + y, P2 de -y et de x,', ...
      ' P1 + 3 bulles P4 de exp(xy), P3 de x^4 + y^4']);
disp(['et de leur derivees : derivees en x, en y et', ...
      ' en xy par projections successives']);

%Choix des types d'espaces : F1, F2, F3, F4 :
D.E = [1,2,14,103] ;

%Choix de l'espace produit E defini par
%E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Choix des formules d'integration sur le domaine
D.intT = 10 ;

%derivations
D.der = [0,1,2] ;

%Definition du maillage et de ses variables T, X, Y, Noeu Front :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;%(facultative ici)
Front = {[7,8,9] ; [9,6,3] ; [3,2,1] ; [1,4,7]} ;%(facultative ici)

%Initialisation des champs de donnees necessaires a EF_Prep
donnees.D = D ;
```

4.3. DESCRIPTION DES FONCTIONS

F^N trace_graphe_interp/proj_deriv_EF

```
%Chargement des coordonnees des DDL (on peut mettre Front = []
%   ci-dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front);

%Calcul de Uc representant la fonction  $f = I_E(g)$  :
%-----%
Ug{1} = Xc{1}' + Yc{1}';
Ug{2} = -Yc{2}';
Ug{3} = Xc{2}';
Ug{4} = exp(Xc{3}' .* Yc{3}');
Ug{5} = Xc{4}' .^4 + Yc{4}' .^4;
Ug = Ug';
Uc = inv_interp_EF_ddl(donnees, Ug, 0, [], X, Y, T, Xc, Yc, Tc);

%Figures dans des tableaux 2x3 de figures :
%-----%

%Figure 1, tableau des graphes des projections des 5 fonctions :
nfig.n = 1;
nfig.t = [2,3];
der{1} = 0;
der{2} = 0;
der{3} = 0;
der{4} = 0;
der{5} = 0;

r = 1 : 5;
%Calcul de Ass_Dom et derivation (d'ordre 0 ici) :
[Vc,SL_Dom] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                             [], X, Y, T);

%Figure 2, tableau des graphes des derivees en x par projection
%   des 5 fonctions :
nfig.n = 2;
```

```

der{1} = 1 ;
der{2} = 1 ;
der{3} = 1 ;
der{4} = 1 ;
der{5} = 1 ;
[Vc2,SL_Dom2] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                               SL_Dom, X, Y, T) ;

%Figure 3, tableau des graphes des derivees en y par projection
%  des 5 fonctions :
nfig.n = 3 ;
der{1} = 2 ;
der{2} = 2 ;
der{3} = 2 ;
der{4} = 2 ;
der{5} = 2 ;
[Vc3,SL_Dom3] = proj_deriv_EF(donnees, Uc, der, r, nfig, ...
                               SL_Dom, X, Y, T) ;

%Figure 4, tableau des graphes des derivees en xy par projection
%  des 5 fonctions :
nfig.n = 4 ;
der{1} = [1,2] ;
der{2} = [1,2] ;
der{3} = [1,2] ;
der{4} = [1,2] ;
der{5} = [1,2] ;
[Vc4,SL_Dom4] = proj_deriv_EF(donnees, Uc, der, r, nfig , ...
                               SL_Dom, X, Y, T) ;

%Trace :
figure(1)
title(['Projection P1 de x+y, P2 de -y et de x,', ...
      ' P1 + bulles P4 de exp(xy), P3 de x^4 + y^4'], ...
      'Position', [-0.7,-1,1]);

```

```

figure(2)
title(['Derivees en x : Pr. P1 x+y, P2 de -y et de x,', ...
      ' P1 + bulles P4 de exp(xy), P3 de x^4 + y^4'], ...
      'Position', [-1.02,-1.3,0]);

figure(3)
title(['Derivees en y : Pr. P1 de x+y, P2 de -y et de x,', ...
      ' P1 + bulles P4 de exp(xy), P3 de x^4 + y^4'], ...
      'Position', [-1.02,-1.3,0]);

figure(4)
title(['Derivees en xy : Pr. P1 de x+y, P2 de -y et de x,', ...
      ' P1 + bulles P4 de exp(xy), P3 de x^4 + y^4'], ...
      'Position', [-1.02,-1.3,0]);

warning on;

```

Algorithme :

EF/trace_graphe_interp/proj_deriv_EF.m

On remet ici la partie déclarative de la fonction :

```

function [Vc, varargout] = proj_deriv_EF(donnees, Uc, der, r, ...
                                         nfig, varargin);

```

- Tests sur les variables et gestions des variables optionnelles. Les variables **der**, **r**, **nfig**, **SL_Dom**, **X**, **Y**, **T**, **Tc**, **Ed** sont alors initialisées par chargement, calcul ou variables d'entrée. La variable **SL_Dom** est soit égale à [] soit est déjà initialisée.
- Initialisation de **Vc** à **Uc**

```
Vc = Uc;
```
- Valeur par défaut de **r** si **r** = [] en entrée :

```
if isempty(r)
    r=1 :length(Uc)
end;
```
- Calcul de **SL_Dom** le cas échéant :

```
if isempty(SL_Dom)
    D = donnees.D;%D.E et D.f description des espaces EF
    SL_Dom = cell(length(D.E),length(D.E));
    for d1=1 :length([0,1,2]);%derivations
        for d2=1 :length([0,1,2]);%derivations
```

```

        for i=1 :length(D.E)
            for j=1 :length(D.E)
                SL_Dom{i,j,d1,d2} = sparse(Ed(i),Ed(j)) ;
            end ;
        end ;
    end ;
end ;
%Assemblage : boucle sur les triangles
s = size(T) ; %T description des sommets du maillage
%Compteurs
f_Dom = cell(length(D.E),length(D.E)) ;
Ass_Dom = [] ;
for i=1 :s(1)
    [Ass_Dom, SL_Dom, f_Dom] = ...
        matrice_elem_prepa_T(donnees, i, T, Tc, X, Y, D, ...
                            Ass_Dom, SL_Dom, f_Dom) ;
end ;
end ;
• Calcul de la projection.
for i=r
    %Derivation par rapport a der{i}
    for j = 1 :length(der{i})
        %Derivee par rapport a der{i}(j)
        %Type d'espace
        p = donnees.D.f(i) ;
        Vc{i} = ...
            SL_Dom{p,p,1,1}\(SL_Dom{p,p,1 + der{i}(j),1}*Vc{i}) ;
    end ;
end ;
• Tracés.
• Variable optionnelle de sortie :
varargout{1} = SL_Dom ;

```


Fonction :

```
function [Vc, varargout] = proj_deriv_EF2(donnees, donnees2, ...
                                         Uc, der, der2, r, nfig, varargin) ;
```

Calcul de dérivées par interpolation d'un espace EF produit $E = E_1 \times \dots \times E_i \times \dots \times E_n$ sur un second espace EF produit $\mathcal{E} = \mathcal{E}_1 \times \dots \times \mathcal{E}_i \times \dots \times \mathcal{E}_n$, puis projection orthogonale de la dérivée de l'interpolée sur $\mathcal{E}$. Les maillages des deux espaces d'EF sont supposés identiques.

La commande

```
Vc = proj_deriv_EF2(donnees, donnees2, Uc, der, der2, ...
                    [], nfig) ;
```

calcule la projection de l'espace EF produit E défini par la variable **donnees** sur le deuxième espace EF produit $\mathcal{E}$ défini par la variable **donnees2**. Soient E_p le p -ième espace de l'espace EF produit E et $\mathcal{E}_p$ le p -ième espace de l'espace EF produit $\mathcal{E}$. Pour une fonction f on note $I_p(f)$ la fonction de $\mathcal{E}_p$ qui interpole f aux *ddl* de $\mathcal{E}_p$.

Soit V la fonction de E_p de composantes $Uc\{p\}$ dans la base de E_p . On alors les deux cas suivants, selon les valeurs prises par les vecteurs de cellules **der** et **der2**.

Cas 1. Les termes de **der** $\{i\}$ et **der2** $\{i\}$ sont des scalaires à valeurs dans $\{0, 1, 2\}$. Alors **Vc** est un vecteur colonne de type CELL tel que **Vc** $\{p\}$ est la composante dans la base de $\mathcal{E}_p$ de la fonction V_e solution de :

$\exists V_e \in \mathcal{E}_p$ tel que :

$$\int_{\Omega} V_e W = \int_{\Omega} I_p(V)_{,der\{p\}} W_{,der2\{p\}} \quad \forall W \in \mathcal{E}_p.$$

Pour **der2** = [] ou 0 (cas usuel), on a alors $V_e = I_p(V)$ si **der** $\{i\} = 0$ et V_e est une approximation de $V_{,x}$ si **der** $\{i\} = 1$ et de $V_{,y}$ si **der** $\{i\} = 2$.

Cas 2. Les termes de **der** $\{i\}$ et de **der2** $\{i\}$ sont des vecteurs de même taille dont les composantes sont à valeurs dans $\{0, 1, 2\}$.

Dans ce cas, V_e sera alors solution des projections successives :

$$V_e^0 = I_p(V),$$

Pour $k = 1, \dots, d = \text{der}\{i\}(\text{end})$, V_e^k est la solution de

$$\exists V_e \in \mathcal{E}_p, \int_{\Omega} V_e^k W = \int_{\Omega} V_e^{k-1} W_{,der\{i\}(k)} \quad \forall W \in \mathcal{E}_p,$$

$$V_e = V_e^d.$$

Pour **der** $\{i\} = 0$ la projection est effectivement calculée et **Vc** $\{i\}$ est simplement l'interpolé de V sur $\mathcal{E}_p$, pour **der** $\{i\} = \text{der2}\{i\} = []$, il y a uniquement interpolation. Le résultat est identique dans les deux cas à la précision machine près si l'intégration est exacte. Les calculs sont effectués à l'erreur d'intégration près.

Le calcul des composantes dans $\mathcal{E}_p$ de $I_p(f)$ se fait en calculant la valeur de cette fonction aux *ddl* de $\mathcal{E}_p$ par la fonction `trac_sol_coup`, puis en calculant ses composantes dans l'espace $\mathcal{E}$ avec `inv_interp_EF_ddl`.

Les champs de la variable `donnees` utilisés sont

```
donnees.D.E,
donnees.D.f,
donnees.D.der, (doit valoir [0,1,2])
donnees.nom_mail, (pour X, Y, T),
donnees.nom_prep, (pour SL_Dom, Xc, Yc, Tc).
```

La variable `donnees.nom_mail` est le nom du fichier du maillage, ce fichier doit contenir les variables X, Y et T, la variable `donnees.nom_prep` est le nom du fichier de données relatif aux espaces d'EF et aux formules de quadrature, il doit contenir la variable `SL_Dom` (pour les calculs de projection cette variable n'est utilisée que si `donnees2` est vide, car alors on considère que `donnees2 = donnees` et `SL_Dom2` ci-dessous prend la valeur `SL_Dom`, sauf si `SL_Dom2` est placée en quinzième variable d'entrée) ainsi que les variables de *ddl* Xc, Yc et Tc.

Ensuite les champs de la variable `donnees2` utilisés sont

```
donnees2.D.E,
donnees2.D.f,
donnees2.D.intT,
donnees2.D.der (doit valoir [0,1,2]).
```

Le rôle de `nfig` est identique à celui joué dans la fonction `proj_deriv_EF`.

La commande

```
[Vc,SL_Dom] = proj_deriv_EF2(donnees, donnees2, Uc, ...
                               der, der2, [], nfig, []);
```

effectue un recalcul de la variable `SL_Dom`. Le champ `donnees.nom_prep` est alors facultatif.

Dans les deux commandes suivantes

```
Vc = proj_deriv_EF2(donnees, donnees2, Uc, der, der2, ...
                    [], nfig, [], X, Y, T);
```

et

```
Vc = proj_deriv_EF2(donnees, donnees2, Uc, der, der2, ...
                    [], nfig, [], X, Y, T, Xc, Yc, Tc);
```

les champs de la variable `donnees` utilisés pour les calculs deviennent

```
donnees.D.E,
donnees.D.f,
```

`donnees.D.der,`

sans les fichiers indiqués par `donnees.nom_prep` et `donnees.nom_mail`.

La variable `SL_Dom2` équivaut pour $\mathcal{E}$ de la variable `SL_Dom` pour E est la troisième variable optionnelle de `proj_deriv_EF2`, on l'obtient ci-dessus avec des appels du type

```
[Vc,SL_Dom,SL_Dom2] = proj_deriv_EF2(...);
```

Enfin la commande

```
Vc = proj_deriv_EF2(donnees, donnees2, Uc, der, der2, ...
                    [], nfig, [], X, Y, T, Xc, Yc, Tc, SL_Dom2);
```

utilise la variable `SL_Dom2`. Cette commande sera utilisée pour des calculs répétitifs pour plus de rapidité.

Les champs de la variable `donnees2` utilisés deviennent alors

```
donnees2.D.E,
donnees2.D.f,
donnees2.D.der.
```

Si `r` est une matrice de 2 lignes, les commandes du type

```
[Vc,SL_Dom,SL_Dom2,d,Vb] = proj_deriv_EF2(donnees, ...
      donnees2, Uc, der, der2, r, nfig, ...);
```

calculent la projection d'un espace EF produit E' sur un deuxième espace EF produit $\mathcal{E}'$ avec

$$E' = E_{r(1,1)} \times \dots \times E_{r(1,end)} \text{ sur } \mathcal{E}' = \mathcal{E}_{r(2,1)} \dots \times \mathcal{E}_{r(2,end)}.$$

En pratique utilisera cette commande avec `r = []` pour avoir $E = E'$ et $\mathcal{E} = \mathcal{E}'$ ou bien `r(1,:) = [1, ..., n]` et les termes de la seconde ligne de `r` égale au même nombre i pour projeter sur $\mathcal{E}_i^n$. Si un terme figure 2 fois sur la ligne 1 de `r`, les termes correspondants de la seconde ligne doivent être identiques.

La variable `Vb` est alors la projection des composantes `r(1,:)` de la fonction représentée par `Uc` et la variable `d` représente avec ses champs `d.D.E` et `d.D.f` l'espace $\mathcal{E}'$:

```
d.D.der = donnees2.D.der ;
d.D.intT = donnees2.D.intT ;
d.D.E = donnees2.D.E ;
d.D.f = donnees2.D.f(r(2, :));
```

C'est elle qui est tracée. Si i n'est pas une composante de `r(1,:)` on a

$$Vc\{i\} = Uc\{i\}.$$

et sinon on a la relation :

$$\forall b\{r(1,i)\} = \forall c\{i\}.$$

Poser `r = [1 : length(Uc); 1 : length(Uc)]` est alors équivalent à poser `r = []`.

On pourra poser `donnees2 = []`, qui est équivalent à `donnees2 = donnees`, sinon les variables `donnees.D.f` et `donnees2.D.f` doivent être de même longueur.

On pourra consulter le `help` de cette fonction pour le rôle des variables.

Exemples :

Exemple 1 : Cet exemple est tiré du `help` de la fonction. On reprend l'exemple donné dans la description de la fonction qui traite le cas de la solution du système à deux inconnues calculé par l'exemple donné par le fichier *exemple_systeme_Bord_cell.m*. Le système (découplé) sur le domaine $\Omega =]0,1[^2$ de frontière notée Γ est :

$$-\Delta u_1 = 2x(1-x) + 2y(1-y), \quad u_1 = 0 \text{ sur } \Gamma,$$

$$-\Delta u_2 = 2x(1-x) + 2y(1-y), \quad u_2 = 0 \text{ sur } \Gamma,$$

de solution

$$u_1(x, y) = u_2(x, y) = x(1-x)y(1-y).$$

Le domaine Ω est muni du maillage du carré unité par 8 triangles de la figure 4.3-5 et on a posé comme espace EF produit E pour l'approximation du problème : $E = E_1 \times E_2$, où l'on a pris E_1 et E_2 égaux à $F_1 = P_4^c$, l'espace d'EF P_4 continu.

Ci-dessous, dans les deux premiers appels de la fonction `proj_deriv_EF2` on a posé `donnees2 = donnees`. Ces deux appels donnent le tracé des projections de u_1 puis de $u_{1,x}$ et $u_{2,y}$.

La projection s'effectue alors de l'espace EF produit $E = [P_4]^2$ continu sur lui même.

Dans le troisième appel de la fonction `proj_deriv_EF` la projection s'effectue de l'espace EF produit $E = [P_4]^2$ continu sur l'espace EF produit $P_3 \times P_4$ discontinu avec la valeur `donnees2.D.E = [103, 104]`.

La projection sur le P_3 discontinu, ne donne pas pas la dérivée exacte à cause du passage par l'interpolation. On pourra utiliser la fonction `proj_deriv_EF3` qui n'utilise pas l'interpolation pour obtenir une dérivée exacte pour ce cas (*cf.* l'exemple 1 de la description de cette fonction qui reprend l'exemple ci-dessous). L'exemple du `help` de la fonction `proj_deriv_EF2` est le suivant :

EXEMPLE 1

```
%Traitement d'une solution EF : celle calculee par
% le fichier exemple_systeme_Bord_cell.
```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF2

```
% On utilise le fichier de nom donnees.nom_prep calcule par
% exemple_systeme_Bord_cell pour recuperer la matrice
% EF SL_Dom.
% On utilise le fichier de nom donnees.nom_mail calcule par
% exemple_systeme_Bord_cell pour recuperer le maillage et
% effectuer les traces.
exemple_systeme_Bord_cell
close all
der{1} = [0] ;
der2{1} = [0] ;
disp('Trace de solution du probleme (2 figures)');
r=[1;1] ;
[Vc1,SL_Dom1,SL_Dom1b,d1,Vb1] = proj_deriv_EF2(donnees, ...
                                                donnees, Uc, der, der2, r, 1) ;

der{1} = [1] ;
der{2} = [1] ;
der2{1} = 0 ;
der2{2} = 0 ;
disp('Derivees par rapport a x (2 fonctions, 2 figures)')
r=[1,2;1,2] ;
[Vc2,SL_Dom2,SL_Dom2b,donnees2b,VVc2] = ...
    proj_deriv_EF2(donnees, donnees, Uc, der, der2, [], 2) ;
%Projection sur le P3 x P4 discontinu (en P3 perte a
% l'interpolation)
%Definition de l'espace :
donnees2.D.der = [0,1,2] ;
donnees2.D.E = [103,104] ;
donnees2.D.f = [1,2] ;
donnees2.D.intT = 10 ;
r = [1,2;1,2] ;
[Vc3,SL_Dom3,SL_Dom3b,d3,Vb3] = ...
    proj_deriv_EF2(donnees, donnees2, Uc, der, der2, r, 4) ;
%Comparaison :
[Xc, Yc, Tc]= ddl(donnees2,X,Y,T,[])
```

```
disp(max(abs(Vc3{1}' - (1 - 2*Xc{1}).*Yc{1}.*(1 - Yc{1})))) ;
disp(max(abs(Vc3{2}' - (1 - 2*Xc{2}).*Yc{2}.*(1 - Yc{2})))) ;
```

Exemple 2 : Cet exemple est mis en œuvre par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple_proj_deriv_EF2 ;
```

Ce fichier reprend l'exemple 2 donné dans la description de la fonction `trac_solEc` où l'espace EF produit E est donné par le maillage du carré unité par 8 triangles de la figure 4.3-5 et

$$E = P_1^c \times P_2^c \times P_2^c \times P_{1,4}^c \times P_3^d,$$

avec $P_{1,4}^c$ l'espace d'EF $P_1 \oplus 3$ bulles P_4 .

On calcule et on trace les composantes de la projection $[L^2(\Omega)]^5$ de $f = (f_1, f_2, f_3, f_4, f_5) \in E$ donnée par :

$$\begin{aligned} f_1(x, y) &= x + y, f_2(x, y) = -y, f_3(x, y) = x^2 + xy, f_4(x, y) = I_{1,4}(e^{xy}), \\ f_5(x, y) &= I_3(x^4 + y^4) \end{aligned}$$

et de leur dérivées

$$f_{i,x}, f_{i,y}, f_{i,xy} \text{ avec } 1 \leq i \leq 5,$$

où pour une fonction f continue $I_{1,4}(f)$ est la fonction qui interpole f aux *ddl* du $P_{1,4}^c$ et $I_3(f)$ est la fonction du P_3^d qui interpole f aux *ddl* du P_3^d . On calcule aussi la dérivation correspondant à

$$\int_{\Omega} V_e W = \int_{\Omega} I_p(V) W_{,y} \quad \forall W \in \mathcal{E}_p.$$

Dans cet exemple pour calculer les composantes de f dans la base EF de l'espace E , on utilise la fonction `inv_interp_EF_ddl` qui résout le problème linéaire d'interpolation à partir des valeurs de g aux *ddl* de E .

On utilise un premier espace de projection qui est l'espace $\mathcal{E} = [P_5^c]^5$. Les valeurs prises par `donnees` sont

`donnees.D`

```
E : [1 2 14 103]
```

```
f : [1 2 2 3 4]
```

```
intT : 10
```

```
der : [0 1 2]
```

et celle prise par `donnees2` qui définissent l'espace sur lequel on projette :

`donnees2.D`

```
E : 105
```

```
f : [1 1 1 1 1]
```

```
intT : 10
```

```
der : [0 1 2]
```

sans utilisation des fichiers `donnees.nom_mail` et `donnees.nom_prep`, avec des commandes du type :

```
[Vc,SL_Dom] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
                             der2, r, nfig, [], X, Y, T) ;
```

ou

```
[Vc2,SL_Dom2] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
                                der2, r, nfig, SL_Dom, X, Y, T) ;
```

Pour l'interpolation de fonctions, les *ddl* sont calculés par une commande du type :

```
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;
```

On utilise les valeurs de `r` suivantes :

```
r = [1, 2, 3, 4, 5 ;
     1, 1, 1, 1, 1] ;
```

ou bien

```
r = [1, 2, 3, 4, 5 ;
     1, 2, 3, 4, 5] ;
```

qui donnent les mêmes projections.

Dans le premier cas la projection se fait de E sur l'espace

$$\mathcal{E}_1 \dots \times \mathcal{E}_1$$

et dans le second cas sur l'espace

$$\mathcal{E}_1 \dots \times \mathcal{E}_5$$

qui sont des espaces identiques pour ce choix de $\mathcal{E}$. Un deuxième exemple d'utilisation de cette fonction est donnée avec l'espace

$$\mathcal{E} = \mathcal{E}_1 \times \mathcal{E}_2 \times \mathcal{E}_3 \times \mathcal{E}_4 \times \mathcal{E}_5 = P_5^d \times P_5^d \times P_6^d \times P_4^d \times P_4^d,$$

défini par

```
donnees2 = donnees ;
```

```
donnees2.D.E = [105,106,104] ;
```

```
donnees2.D.f = [1,1,2,3,3] ;
```

On pose comme choix de `r` :

```
r = [] ;
```

pour une projection de E sur $\mathcal{E}$, puis

```
r = [1, 2, 3, 4, 5 ;
      1, 1, 1, 1, 1] ;
```

pour une projection de E sur

$$\mathcal{E}' = \mathcal{E}_1 \times \mathcal{E}_1 \times \mathcal{E}_1 \times \mathcal{E}_1 \times \mathcal{E}_1 = P_5^d \times P_5^d \times P_5^d \times P_5^d \times P_5^d,$$

ensuite

```
r = [1, 2, 3, 4, 5 ;
      1, 2, 3, 1, 4] ;
```

pour une projection de E sur

$$\mathcal{E}' = \mathcal{E}_1 \times \mathcal{E}_2 \times \mathcal{E}_3 \times \mathcal{E}_1 \times \mathcal{E}_4 = P_5^d \times P_5^d \times P_6^d \times P_5^d \times P_4^d,$$

et enfin

```
r = [1, 3, 2, 4, 4 ;
      4, 2, 3, 4, 4] ;
```

pour une projection de la fonction représentée par

$$\{\text{Uc}\{1\}; \text{Uc}\{3\}; \text{Uc}\{2\}; \text{Uc}\{4\}; \text{Uc}\{4\}\}$$

de

$$E = E_1 \times E_3 \times E_2 \times E_4 \times E_4$$

sur l'espace

$$\mathcal{E}' = \mathcal{E}_4 \times \mathcal{E}_2 \times \mathcal{E}_3 \times \mathcal{E}_4 \times \mathcal{E}_4 = P_4^d \times P_5^d \times P_6^d \times P_4^d \times P_4^d,$$

Cette projection est effectuée, comme pour les opérations précédentes, par une commande du type

```
[Vc9, SL_Dom9, SL_Dom2d, d9, Vb9] = proj_deriv_EF2(donnees, ...
              donnees2, Uc, der, der, r, 'n', [], X, Y, T) ;
```

Le résultat est mis dans le quatrième variable de sortie `Vc9` qui contient la projection. La variable `d9` définit l'espace $\mathcal{E}'$:

```
der : [0 1 2]
intT : 10
E : [105 106 104]
f : [3 1 2 3 3]
```

Pour `Vc9` on a aussi $\text{Vc9}\{r(1, i)\} = \text{Vb9}\{i\}$ et $\text{Vc9}\{5\} = \text{Uc}\{5\}$ ce qui donne

$$\text{Vc9} = \{\text{Vb9}\{1\}; \text{Vb9}\{3\}; \text{Vb9}\{2\}; \text{Vb9}\{4\}; \text{Uc}\{5\}\}$$

pour une projection de la fonction représentée par

$$\{\text{Uc}\{1\}; \text{Uc}\{2\}; \text{Uc}\{3\}; \text{Uc}\{4\}; \text{Uc}\{5\}\}$$

de E sur l'espace

$$\mathcal{E}_4 \times \mathcal{E}_3 \times \mathcal{E}_2 \times \mathcal{E}_4 \times E_5 = P_4^d \times P_6^d \times P_5^d \times P_4^d \times P_3^d.$$

Les quatre premiers espaces sont aussi définis par $\text{d9.D.E} = [105, 106, 104]$ pour les types d'espaces et $\text{d9.D.f}(r(1, :)) = [3, 2, 1, 3, 3]$ pour les espaces.

Le fichier *script* est le suivant :

EF/exemples/trace_graphe_interp/exemple_proj_deriv_EF2.m

```
%Fichier EF/trace_graphe_interp/exemple_proj_deriv_EF2.m
%Utilisation de PROJ_DERIV_EF2. Projections L2 et traces de
% fonctions et de leur derivees d'un espace d'EF E sur un
% espace G avec E = E1 x E2 x E3 x E4 x E5 :
%     E1 = P1 continu = F1,
%     E2 = P2 ontinu = F2,
%     E3 = P2 continu = F2,
%     E4 = P1 + 3 bulle P4 continu = F3,
%     E5 = P3 discontinu = F4.
%
% La projection est effectuee sur G = G1 x G2 x G3 x G4 x G5
% avec G1 = G2 = G3 = G4 = G5 = H1 = P5 discontinu, un trace
% de figures est aussi effectuee.
%
% Les calculs sont effectues sur l'interpolee sur E de la
% fonction g definie par :
%
%           g = (x + y, -y, x, exp(xy), x^4 + y^4).
%
% La fonction INV_INTERP_EF_DDL calcule les composantes Uc de
% la fonction f = I_E(g) element de E qui vaut g aux DDL de E.
%
% Un second exemple de projection est aussi donne avec G donne
% par :
%
%     G1 = P5 discontinu = H1,
%     G2 = P5 discontinu = H1,
%     G3 = P6 discontinu = H2,
%     G4 = P4 discontinu = H3,
%     G5 = P4 discontinu = H3
%
% et differentes valeurs de r :
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/proj_deriv_EF2

```
%
%  r = [] pour une projection de E sur G,
%
%  r = [1, 2, 3, 4, 5 ;
%       1, 1, 1, 1, 1] ;
%
%  pour une projection de E1 x E2 x E3 x E4 x E5
%  sur                  G1 x G1 x G1 x G1 x G1,
%
%  r = [1, 2, 3, 4, 5 ;
%       1, 2, 3, 1, 4] ;
%
%  pour une projection de E1 x E2 x E3 x E4 x E5
%  sur                  G1 x G2 x G3 x G1 x G4,
%
%  r = [1, 3, 2, 4, 4 ;
%       4, 2, 3, 4, 4] ;
%
%  pour une projection de E1 x E2 x E3 x E4 x E4
%  sur                  G4 x G2 x G3 x G1 x G4.

clear ;
close all ;
warning off ;
disp('Exemple1. Projection de E sur G = [P5]^5 discontinu d''une') ;
disp('fonction g et de ses derivees : derivee en x, en y et en ') ;
disp('xy par projections successives et derivee en y sur la') ;
disp('fonction test, avec E = P1c x P2c x P2c x P1,4c x P3d et') ;
disp('g(x,y) = (x + y, -y, x, exp(xy), x^4 + y^4)') ;

%Choix des types d'espaces : F1, F2, F3, F4 :
D.E = [1,2,14,103] ;

%Choix de l'espace produit E defini par
```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF2

```
%E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4] ;

%Choix des formules d'integration sur le domaine
D.intT = 10 ;

%derivations
D.der = [0,1,2] ;

%Calcul du maillage, variables T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;%facultatif

%Dont la frontiere, variable Front (facultative) :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep
donnees.D = D ;

%Chargement des coordonnees des DDL (on peut mettre Front = []
%   ci-dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Espace P5d de projection :
%-----%
donnees2 = donnees ;
%Choix du type d'espace H1 :
donnees2.D.E = 105 ;
```

4. DESCRIPTION DU CODE FONCTION trace_graphe_interp/proj_deriv_EF2

```
%Choix de l'espace produit E defini par
%G = G1 x G2 x G3 x G4 x G5 = H1 x H1 x H1 x H1 x H1 :
donnees2.D.f = [1,1,1,1,1] ;

%Derivation nulle sur la fonction test :
der2 = [] ;

%Calcul de Uc representant la fonction f = I_E(g) :
%-----%
Ug{1} = Xc{1}' + Yc{1}' ;
Ug{2} = -Yc{2}' ;
Ug{3} = Xc{2}' ;
Ug{4} = exp(Xc{3}' .* Yc{3}') ;
Ug{5} = Xc{4}' .^4 + Yc{4}' .^4 ;
Ug = Ug' ;
Uc = inv_interp_EF_ddl(donnees, Ug, 0, [], X, Y, T, Xc, Yc, Tc) ;

%Figures dans des tableaux 2x3 de figures :
%-----%
%Figure 1, tableau des graphes des projections des 5 fonctions :
nfig.n = 1 ;
nfig.t = [2,3] ;

%Derivations :
der{1} = 0 ;
der{2} = 0 ;
der{3} = 0 ;
der{4} = 0 ;
der{5} = 0 ;

%Projection de E sur G' = E1 x E1 x E1 x E1 x E1, sans derivation :
%-----%
r = [1, 2, 3, 4, 5 ;
     1, 1, 1, 1, 1] ;
```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF2

```
[Vca,SL_Doma,SL_Dom2a,da,Vba] = proj_deriv_EF2(donnees, ...
    donnees2, Uc, der, der2, r, nfig, [], X, Y, T);

%Projection de E sur G' = G = E1 x E2 x E3 x E4 x E5 = E1^5 :
%-----%
%Le calcul suivant donne le meme resultat que precedemment :
% On calcule Ass_Dom pour la suite.
r = [1, 2, 3, 4, 5;
    1, 2, 3, 4, 5];

[Vc,SL_Dom,SL_Dom2,d,Vb] = proj_deriv_EF2(donnees, ...
    donnees2, Uc, der, der2, r, nfig, [], X, Y, T);

%4 exemples de projection de E sur G avec derivations :
%-----%
nfig.n = 2;
der{1} = 1;
der{2} = 1;
der{3} = 1;
der{4} = 1;
der{5} = 1;
[Vc2,SL_Dom2] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
    der2, r, nfig, SL_Dom, X, Y, T);

nfig.n = 3;
der{1} = 2;
der{2} = 2;
der{3} = 2;
der{4} = 2;
der{5} = 2;
[Vc3,SL_Dom3] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
    der2, r, nfig, SL_Dom, X, Y, T);

nfig.n = 4;
```

```

der{1} = [1,2] ;
der{2} = [1,2] ;
der{3} = [1,2] ;
der{4} = [1,2] ;
der{5} = [1,2] ;
[Vc4,SL_Dom4] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
                                der2, r, nfig, SL_Dom, X, Y, T) ;

nfig.n = 5 ;
der{1} = [] ;
der{2} = [] ;
der{3} = [] ;
der{4} = [] ;
der{5} = [] ;
%La derivation porte sur la fonction test, notee y''
der2{1} = [2] ;
der2{2} = [2] ;
der2{3} = [2] ;
der2{4} = [2] ;
der2{5} = [2] ;
[Vc5,SL_Dom5] = proj_deriv_EF2(donnees, donnees2, Uc, der, ...
                                der2, r, nfig, SL_Dom, X, Y, T) ;

%Trace :
figure(1)
title(['Projection P5 de x + y (P1), de -y et x (P2),', ...
      ' exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-0.7,-1,1]) ;

figure(2)
title(['Derivees en x : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.02,-1.3,0]) ;

figure(3)
title(['Derivees en y : Pr. P5 de x + y (P1), de -y et', ...

```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF2

```

' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
                                'Position', [-1.02,-1.3,0]);

figure(4)
title(['Derivees en xy : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.075,-1.37,0]);

figure(5)
title(['Derivees en y'' : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.02,-1.3,0]);

disp('Exemple2. Proj. sur le P5 x P5 x P6 x P4 x P4 discontinu');
%Projection sur le P5 x P5 x P6 x P4 x P4 discontinu :
%-----%
%Choix des types d'espaces : H1, H2, H3 :
donnees2 = donnees;
donnees2.D.E = [105,106,104];

%Choix de l'espace produit G defini par :
%   G = G1 x G2 x G3 x G4 x G5 = H1 x H1 x H2 x H3 x H3
%       = P5 x P5 x P6 x P4 x P4 discontinu :
donnees2.D.f = [1,1,2,3,3];

%Derivations :
der{1} = 0;
der{2} = 0;
der{3} = 0;
der{4} = 0;
der{5} = 0;
der2 = der;
%Projection de E sur G :
r = [];
[Vc6,SL_Dom6, SL_Dom2a, d6, Vb6] = proj_deriv_EF2(donnees, ...
            donnees2, Uc, der, der, r, 'n', [], X, Y, T);

```

```

%Projection de E sur G' = G1 x G1 x G1 x G1 x G1 = [P5]^5 :
r = [1, 2, 3, 4, 5;
     1, 1, 1, 1, 1];
[Vc7,SL_Dom7, SL_Dom2b, d7, Vb7] = proj_deriv_EF2(donnees, ...
     donnees2, Uc, der, der, r, 'n', [], X, Y, T);

%Projection de E sur G' = G1 x G2 x G3 x G1 x G4
%                               = P5 x P5 x P6 x P5 x P4 :
r = [1, 2, 3, 4, 5;
     1, 2, 3, 1, 4];
[Vc8, SL_Dom8, SL_Dom2c, d8, Vb8] = proj_deriv_EF2(donnees, ...
     donnees2, Uc, der, der, r, 'n', [], X, Y, T);

%Projection de E' = E1 x E2 x E3 x E4 x E4 sur
%   G' = G4 x G2 x G3 x G4 x G4 = P4 x P5 x P5 x P4 x P4
%   (ici 5 ne figure pas sur la ligne de 1 de r) :
r = [1, 3, 2, 4, 4;
     4, 2, 3, 4, 4];%compatibilite colonnes 4 et 5

%On a Vc9{5} = Uc{5} et Vb9 est la projection de la fonction de E'
%   representee par [Uc{1},Uc{3},Uc{2},Uc{4},Uc{4}] :
[Vc9, SL_Dom9, SL_Dom2d, d9, Vb9] = proj_deriv_EF2(donnees, ...
     donnees2, Uc, der, der, r, 'n', [], X, Y, T);

%L'espace de Vc9 est donne par :
donnees3.D.E = [105, 106, 104, 103];
donnees3.D.f = d9.D.f(r(1, :));
donnees3.D.f(5) = 4;

%Traces de U et des projections :
%Ordre comme [Uc{1},Uc{2},Uc{3},Uc{4},Uc{5}] :
nfig.n = 6;
trac_solEc(donnees, Uc, 0, 1 :5, 6, nfig, [], X, Y, T);

```



```

figure(6) ;
title('Second espace de projection : fonction a projeter', ...
      'Position', [-0.5,-0.65,0]) ;
%Ordre comme [Uc{1},Uc{3},Uc{2},Uc{4},Uc{4}] :
nfig.n = 7 ;
trac_solEc(d9, Vb9, 0, 1 :5, 6, nfig, [], X, Y, T) ;
figure(7) ;
title(['Second espace de projection ordre :', ...
      ' u1, u3, u2, u4, u4'], 'Position', [-0.5,-0.7,0]) ;
%Ordre comme [Uc{1},Uc{2},Uc{3},Uc{4},Uc{5}] :
nfig.n = 8 ;
trac_solEc(donnees3, Vc9, 0, 1 :5, 6, nfig, [], X, Y, T) ;
figure(8)
title(['Second espace de projection ordre : u1, u2, u3,', ...
      ' u4, u5 (non projete)'], ...
      'Position', [-0.5,-0.65,0]) ;

warning on ;

```

Fonction :

```

function [Vc, varargout] = proj_deriv_EF2b(donnees, donnees2, ...
      Uc, der, der2, r, nfig, varargin) ;

```

Calcul de dérivées par interpolation d'un espace EF produit E sur un second espace EF produit $\mathcal{E}$, puis projection orthogonale de la dérivée de l'interpolée sur $\mathcal{E}$.

La fonction `proj_deriv_EF2b` calcule la même valeur de Vc que la fonction `proj_deriv_EF2` précédente. La différence vient de sa syntaxe qui est celle de fonction `proj_deriv_EF3` qui suit. La syntaxe est modifiée par rapport aux calculs des variables `SL_Dom` et `SL_Dom2` ci-dessus, et par rapport variables d'entrée (ordre) et de sortie (ordre et nature des variables de sortie).

Fonction :

```

function [Vc, varargout] = proj_deriv_EF3(donnees, donnees2, ...
      Uc, der, der2, r, nfig, varargin) ;

```

Cette fonction effectue les mêmes opérations que la fonction `proj_deriv_EF2` avec les différences suivantes :

- La syntaxe est modifiée, avec notamment pour la gestion des variables `SL_Dom` et `SL_Dom2`, des différences au niveau des variables d'entrée (ordre) et de sortie (ordre et nature des variables de sortie).

- Les calculs sont effectués non sur l'interpolé de la fonction sur l'espace de projection mais sur la fonction elle-même. Il n'y a alors pas d'erreur d'interpolation sur la fonction, mais une erreur sur l'intégration de termes dépendant de cette fonction

Ainsi l'erreur sur la dérivation de l'exemple 1 donné dans la description de la fonction de `proj_deriv_EF2` est nulle si on utilise la fonction `proj_deriv_EF3` comme dans l'exemple 1 ci-dessous.

Calculs de dérivées par projection orthogonale d'un espace EF produit E sur un second espace EF produit $\mathcal{E}$.

La commande

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, [], nfig, X, Y, T);
```

ou la commande

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, [], nfig, X, Y, T, Xc, Yc, Tc);
```

calculent la projection de l'espace EF produit E défini par la variable `donnees` sur le deuxième espace EF produit $\mathcal{E}$ défini par la variable `donnees2`.

Soient E_p le p -ième espace de l'espace EF produit E et $\mathcal{E}_p$ le p -ième espace de l'espace EF produit $\mathcal{E}$. Soit V la fonction de E_p de composantes $Uc\{p\}$ dans la base de E_p . Alors Vc est un vecteur colonne de type `CELL` tel que $Vc\{p\}$ est la composante dans la base de $\mathcal{E}_p$ de la fonction V_e solution de :

$\exists V_e \in \mathcal{E}_p$ tel que

$$\int_{\Omega} V_e W = \int_{\Omega} V_{,der\{p\}} W_{,der2\{p\}} \quad \forall W \in E_p.$$

Les calculs sont effectués à l'erreur d'intégration près.

Cette fonction est analogue à la fonction `proj_deriv_EF2`, la différence porte sur la syntaxe et sur la valeur de Vc qui, dans cette fonction, est donnée par :

$$\int_{\Omega} V_e W = \int_{\Omega} I_p(V)_{,der\{p\}} W_{,der2\{p\}} \quad \forall W \in E_p,$$

où $I_p(V)$ est la fonction de $\mathcal{E}_p$ qui interpole V aux *ddl* de $\mathcal{E}_p$.

Les dérivations ne sont programmées que pour l'ordre 1 (pas de dérivations successives comme dans `proj_deriv_EF`), on pourra réitérer avec `proj_deriv_EF` ou `proj_deriv_EF3` pour un calcul de dérivées d'ordre supérieur.

4.3. DESCRIPTION DES FONCTIONS F^N `trace_graphe_interp/proj_deriv_EF3`

La fonction `proj_deriv_EF2b` a la même syntaxe que `proj_deriv_EF3`, elle permet le calcul des dérivées successives et calcule la même valeur de V_c que la fonction `proj_deriv_EF2`.

Pour les deux commandes ci-dessus les champs de la variable `donnees` utilisés sont

```
donnees.D.E,
donnees.D.f,
donnees.D.der
```

et ceux de la variable `donnees2` :

```
donnees2.D.E,
donnees2.D.f,
donnees2.D.intT,
donnees2.D.der.
```

Les variables X , Y et T sont les variables de maillage et X_c , Y_c et T_c les variables de *ddl* relatives à l'espace E .

En sortie la variable d décrit par son champ $d.D$ l'espace $\mathcal{E}$ et la variable $V_b = V_c$ la projection sur E . Lorsque r est choisi différent de `[]`, les variables d et V_b sont alors modifiées comme décrit ci-dessous. Les cellules de la variable S contiennent les variables de noms

```
Ass_Dom2, Ed2, SL_Dom2, Tc, Tc2, Xc, Xc2, Yc, Yc2, f_Dom2.
```

qui peuvent être utilisées pour accélérer les calculs, par exemple lorsque ceux-ci sont répétitifs. On utilisera alors la commande :

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, [], nfig);
```

Les champs suivants de la variable `donnees` utilisés sont alors

```
donnees.D.E,
donnees.D.f,
donnees.D.der,
```

et ceux de la variable `donnees2` :

```
donnees2.D.E,
donnees2.D.f,
donnees2.D.intT,
donnees2.D.der,
donnees.nom_mail
donnees.nom_prep.
```

Ci-dessus, la variable `donnees2.D.f.nom_mail` est le nom du fichier du maillage qui doit contenir les variables `X`, `Y` et `T` et la variable `donnees2.D.nom_prep` est le nom du fichier qui doit contenir les variables

`Ass_Dom2, Ed2, SL_Dom2, Tc, Tc2, Xc, Xc2, Yc, Yc2, f_Dom2,`

vues ci-dessus.

Si `r` est une matrice de 2 lignes, les commandes

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, r, nfig);
```

ou

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, r, nfig, X, Y, T);
```

ou

```
[Vc, d, Vb, S] = proj_deriv_EF3(donnees, donnees2, ...
                                Uc, der, der2, r, nfig, X, Y, T, Xc, Yc, Tc);
```

calculent la projection d'un espace EF produit E' sur un deuxième espace EF produit $\mathcal{E}'$ avec

$$E' = E_{r(1,1)} \times \dots \times E_{r(1,end)} \text{ sur } \mathcal{E}' = \mathcal{E}_{r(2,1)} \dots \times \mathcal{E}_{r(2,end)}.$$

En pratique on pourra choisir $E = E'$ et $\mathcal{E} = \mathcal{E}'$.

En sortie la variable `d` décrit par ses champs `d.D.E` et `d.D.f` l'espace $\mathcal{E}'$ et la variable `Vb` la projection sur $\mathcal{E}'$ de la fonction formée des composantes `r(1,:)` de la fonction représentée par `Uc`.

Si i n'est pas une composante de `r(1,:)` alors

$$Vc\{i\} = Uc\{i\}.$$

et sinon

$$Vc\{r(1,i)\} = Vb\{i\}.$$

Poser `r = [1 : length(Uc); 1 : length(Uc)]` est alors équivalent à poser `r = []` ;

On pourra consulter le `help` de cette fonction pour le rôle des variables.

Exemples :

Exemple 1 : Cet exemple est tiré du `help` de la fonction. On reprend l'exemple donné dans la description de la fonction qui traite le cas de la solution du système à deux inconnues calculé par l'exemple donné par le fichier `exemple_systeme_Bord_cell.m`. Le système (découplé) sur le domaine $\Omega =]0,1[^2$ de frontière notée Γ est :

$$-\Delta u_1 = 2x(1-x) + 2y(1-y), \quad u_1 = 0 \text{ sur } \Gamma,$$

$$-\Delta u_2 = 2x(1-x) + 2y(1-y), \quad u_2 = 0 \text{ sur } \Gamma,$$

de solution

$$u_1(x, y) = u_2(x, y) = x(1-x)y(1-y).$$

Le domaine Ω est muni du maillage du carré unité par 8 triangles de la figure 4.3-5 et on a posé comme espace EF produit E pour l'approximation du problème : $E = E_1 \times E_2$, où l'on a pris E_1 et E_2 égaux à $F_1 = P_4^c$, le P_4 continu.

Ci-dessous, dans les deux premiers appels de la fonction `proj_deriv_EF3` on a posé `donnees2 = donnees`. Ces deux appels donnent le tracé des projections de u_1 puis de $u_{1,x}$ et $u_{2,y}$.

La projection s'effectue alors de l'espace EF produit $E = [P_4]^2$ continu sur lui même.

Dans le troisième appel de la fonction `proj_deriv_EF3` la projection s'effectue de l'espace EF produit $E = P_4 \times P_4$ continu sur l'espace EF produit $P_3 \times P_4$ discontinu avec la valeur `donnees2.D.E = [103,104]`. On obtient alors dans les deux cas la dérivée exacte. L'exemple du `help` de la fonction `proj_deriv_EF3` est le suivant :

EXEMPLE 1

```
%Traitement d'une solution EF : celle calculee par
%   le fichier exemple_systeme_Bord_cell.
%   On utilise le fichier de nom donnees.nom_prep calcule par
%   exemple_systeme_Bord_cell pour recuperer la matrice
%   EF SL_Dom.
%   On utilise le fichier de nom donnees.nom_mail calcule par
%   exemple_systeme_Bord_cell pour recuperer le maillage et
%   effectuer les traces.
exemple_systeme_Bord_cell
close all
der{1} = [0] ;
der2{1} = [0] ;
disp('Trace de solution du probleme (2 figures)');
r=[1;1] ;
[Vc,SL_Dom] = proj_deriv_EF3(donnees, donnees, Uc, der, ...
                             der2, r, 1, X, Y, T) ;

der{1} = [1] ;
der{2} = [1] ;
```

```

der2{1} = 0 ;
der2{2} = 0 ;
disp('Derivees par rapport a x (2 fonctions, 2 figures)')
r=[1,2;1,2] ;
[Vc2,SL_Dom2] = proj_deriv_EF3(donnees, donnees, Uc, der, ...
                                der2, [], 2, X, Y, T) ;
%Projection sur le P3 x P4 discontinu (en P3 pas de perte ici)
%Definition de l'espace :
donnees2.D.der = [0,1,2] ;
donnees2.D.E = [103,104] ;
donnees2.D.f = [1,2] ;
donnees2.D.intT = 10 ;
r = [1,2;1,2] ;
[Vc3,SL_Dom3] = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                                der2, r, 4, X, Y, T) ;

%Comparaison :
[Xc, Yc, Tc]= ddl(donnees2,X,Y,T,[]) ;
disp(max(abs(Vc3{1}' - (1 - 2*Xc{1}).*Yc{1}.*(1 - Yc{1})))) ;
disp(max(abs(Vc3{2}' - (1 - 2*Xc{2}).*Yc{2}.*(1 - Yc{2})))) ;

```

Exemple 2 : Cet exemple est mis en œuvre par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple_proj_deriv_EF3 ;
```

il est analogue à l'exemple 2 décrit à la fonction `proj_deriv_EF2`. Le fichier *script* est le suivant :

EF/exemples/trace_graphe_interp/exemple_proj_deriv_EF3.m

```

%Fichier EF/trace_graphe_interp/exemple_proj_deriv_EF3.m
%Utilisation de PROJ_DERIV_EF3. Projections L2 et traces de
% fonctions et de leur derivees d'un espace d'EF E sur un
% espace G avec E = E1 x E2 x E3 x E4 x E5 :
%     E1 = P1 continu = F1,
%     E2 = P2 ontinu = F2,
%     E3 = P2 continu = F2,
%     E4 = P1 + 3 bulle P4 continu = F3,
%     E5 = P3 discontinu = F4.

```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF3

```
%
% La projection est effectuee sur  $G = G_1 \times G_2 \times G_3 \times G_4 \times G_5$ 
% avec  $G_1 = G_2 = G_3 = G_4 = G_5 = H_1 = P_5$  discontinu, un trace
% de figures est aussi effectuee.
%
% La projection est effectuee sur  $G = G_1 \times G_2 \times G_3 \times G_4 \times G_5$ 
% avec  $G_1 = G_2 = G_3 = G_4 = G_5 = H_1 = P_5$  discontinu, un trace
% de figures est effectuee.
%
% Les calculs sont effectues sur l'interpolee sur E de la
% fonction g definie par :
%
%           $g = (x + y, -y, x, \exp(xy), x^4 + y^4).$ 
%
% La fonction INV_INTERP_EF_DDL calcule les composantes  $U_c$  de
% la fonction  $f = I_E(g)$  element de E qui vaut g aux DDL de E.
%
% Un second exemple de projection est aussi donne avec G donne
% par :
%
%           $G_1 = P_5 \text{ discontinu} = H_1,$ 
%           $G_2 = P_5 \text{ discontinu} = H_1,$ 
%           $G_3 = P_6 \text{ discontinu} = H_2,$ 
%           $G_4 = P_4 \text{ discontinu} = H_3,$ 
%           $G_5 = P_4 \text{ discontinu} = H_3$ 
%
% et differentes valeurs de r :
%
% r = [] pour une projection de E sur G,
%
% r = [1, 2, 3, 4, 5;
%      1, 1, 1, 1, 1];
%
% pour une projection de  $E_1 \times E_2 \times E_3 \times E_4 \times E_5$ 
```

4. DESCRIPTION DU CODE

FONCTION trace_graphe_interp/proj_deriv_EF3

```
% sur G1 x G1 x G1 x G1 x G1,
%
% r = [1, 2, 3, 4, 5;
%      1, 2, 3, 1, 4];
%
% pour une projection de E1 x E2 x E3 x E4 x E5
% sur G1 x G2 x G3 x G1 x G4,
%
% r = [1, 3, 2, 4, 4;
%      4, 2, 3, 4, 4];
%
% pour une projection de E1 x E2 x E3 x E4 x E4
% sur G4 x G2 x G3 x G1 x G4.

clear;
close all;
warning off;
disp('Exemple1. Projection de E sur G = [P5]^5 discontinu d''une');
disp('fonction g et de ses derivees : derivee en x, en y et en ');
disp('xy par projections successives et derivee en y sur la');
disp('fonction test, avec E = P1c x P2c x P2c x P1,4c x P3d et');
disp('g(x,y) = (x + y, -y, x, exp(xy), x^4 + y^4)');

%Choix des types d'espaces : F1, F2, F3, F4 :
D.E = [1,2,14,103];

%Choix de l'espace produit E defini par
%E = E1 x E2 x E3 x E4 x E5 = F1 x F2 x F2 x F3 x F4 :
D.f = [1,2,2,3,4];

%Choix des formules d'integration sur le domaine
D.intT = 10;

%derivations
```


4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF3

```

D.der = [0,1,2] ;

%Calcul du maillage, variables T, X, Y :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ; 4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Noeu = length(X) ;%facultatif

%Dont la frontiere, variable Front (facultative) :
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;

%Initialisation des champs de donnees necessaires a EF_Prep
donnees.D = D ;

%Chargement des coordonnees des DDL (on peut mettre Front = []
%   ci-dessous) :
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front) ;

%Espace P5d de projection :
%-----%
donnees2 = donnees ;
%Choix du type d'espace H1 :
donnees2.D.E = 105 ;

%Choix de l'espace produit E defini par
%G = G1 x G2 x G3 x G4 x G5 = H1 x H1 x H1 x H1 x H1 :
donnees2.D.f = [1,1,1,1,1] ;

%Derivation nulle sur la fonction test :
der2 = [] ;

```

```

%Calcul de Uc representant la fonction  $f = I_E(g)$  :
%-----%
Ug{1} = Xc{1}' + Yc{1}';
Ug{2} = -Yc{2}';
Ug{3} = Xc{2}';
Ug{4} = exp(Xc{3}' .* Yc{3}');
Ug{5} = Xc{4}' .^4 + Yc{4}' .^4;
Ug = Ug';
Uc = inv_interp_EF_ddl(donnees, Ug, 0, [], X, Y, T, Xc, Yc, Tc);

%Figures dans des tableaux 2x3 de figures :
%-----%
%Figure 1, tableau des graphes des projections des 5 fonctions :
nfig.n = 1;
nfig.t = [2,3];

%Derivations :
der{1} = 0;
der{2} = 0;
der{3} = 0;
der{4} = 0;
der{5} = 0;

%Projection de E sur  $G' = E_1 \times E_1 \times E_1 \times E_1 \times E_1$ , sans derivation :
%-----%
r = [1, 2, 3, 4, 5;
     1, 1, 1, 1, 1];
Vca = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                    der2, r, nfig, X, Y, T);

%Projection de E sur  $G' = G = E_1 \times E_2 \times E_3 \times E_4 \times E_5 = E_1^5$  :
%-----%
%Le calcul suivant donne le meme resultat que precedemment :
r = [1, 2, 3, 4, 5;

```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF3

```

    1, 2, 3, 4, 5] ;
Vc = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                    der2, r, nfig, X, Y, T) ;

%4 exemples de projection de E sur G avec derivations :
%-----%
nfig.n = 2 ;
der{1} = 1 ;
der{2} = 1 ;
der{3} = 1 ;
der{4} = 1 ;
der{5} = 1 ;
Vc2 = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                    der2, r, nfig, X, Y, T) ;

nfig.n = 3 ;
der{1} = 2 ;
der{2} = 2 ;
der{3} = 2 ;
der{4} = 2 ;
der{5} = 2 ;
Vc3 = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                    der2, r, nfig, X, Y, T) ;

nfig.n = 4 ;
der{1} = [2] ;
der{2} = [2] ;
der{3} = [2] ;
der{4} = [2] ;
der{5} = [2] ;

Vc4 = proj_deriv_EF3(donnees, donnees2, Vc2, der, ...
                    der2, r, nfig, X, Y, T) ;

```

```

nfig.n = 5 ;
der{1} = [] ;
der{2} = [] ;
der{3} = [] ;
der{4} = [] ;
der{5} = [] ;
der2{1} = [2] ;
der2{2} = [2] ;
der2{3} = [2] ;
der2{4} = [2] ;
der2{5} = [2] ;
Vc5 = proj_deriv_EF3(donnees, donnees2, Uc, der, ...
                    der2, r, nfig, X, Y, T) ;

%Trace :
figure(1)
title(['Projection P5 de x + y (P1), de -y et x (P2),', ...
      ' exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1,-1.4,1]) ;

figure(2)
title(['Derivees en x : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.02,-1.3,0]) ;

figure(3)
title(['Derivees en y : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.02,-1.3,0]) ;

figure(4)
title(['Derivees en xy : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.075,-1.37,0]) ;

figure(5)
title(['Derivees en y'' : Pr. P5 de x + y (P1), de -y et', ...
      ' x (P2), exp(xy) (P1 + bulles P4), x^4 + y^4 (P3)'], ...
      'Position', [-1.02,-1.3,0]) ;

```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF3

```

disp('Exemple2. Proj. sur le P5 x P5 x P6 x P4 x P4 discontinu');
%Projection sur le P5 x P5 x P6 x P4 x P4 discontinu :
%-----%
%Choix des types d'espaces : H1, H2, H3 :
donnees2 = donnees ;
donnees2.D.E = [105,106,104] ;

%Choix de l'espace produit G defini par :
%   G = G1 x G2 x G3 x G4 x G5 = H1 x H1 x H2 x H3 x H3
%       = P5 x P5 x P6 x P4 x P4 discontinu :
donnees2.D.f = [1,1,2,3,3] ;

%Derivations :
der{1} = 0 ;
der{2} = 0 ;
der{3} = 0 ;
der{4} = 0 ;
der{5} = 0 ;
der2 = der ;
%Projection de E sur G :
r = [] ;
[Vc6, d6, Vb6, S6] = proj_deriv_EF3(donnees, ...
                                     donnees2, Uc, der, der, r, 'n', X, Y, T) ;

%Projection de E sur G' = G1 x G1 x G1 x G1 x G1 = [P5]^5 :
r = [1, 2, 3, 4, 5 ;
     1, 1, 1, 1, 1] ;
[Vc7, d7, Vb7, S7] = proj_deriv_EF3(donnees, donnees2, ...
                                     Uc, der, der, r, 'n', X, Y, T) ;

%Projection de E sur G' = G1 x G2 x G3 x G1 x G4
%                         = P5 x P5 x P6 x P5 x P4 :
r = [1, 2, 3, 4, 5 ;

```

4. DESCRIPTION DU CODE FONCTION trace_graphe_interp/proj_deriv_EF3

```

    1, 2, 3, 1, 4] ;
[Vc8, d8, Vb8, S8] = proj_deriv_EF3(donnees, donnees2, ...
    Uc, der, der, r, 'n', X, Y, T) ;

%Projection de E' = E1 x E2 x E3 x E4 x E4 sur
%   G' = G4 x G2 x G3 x G4 x G4 = P4 x P5 x P5 x P4 x P4
%   (ici 5 ne figure pas sur la ligne de 1 de r) :
    r = [1, 3, 2, 4, 4;
        4, 2, 3, 4, 4] ; %compatibilite colonnes 4 et 5

%On a Vc9{5} = Uc{5} et Vb9 est la projection de la fonction de E'
%   representee par [Uc{1},Uc{3},Uc{2},Uc{4},Uc{4}] :
[Vc9, d9, Vb9, S9] = proj_deriv_EF3(donnees, donnees2, ...
    Uc, der, der, r, 'n', X, Y, T) ;

%L'espace de Vc9 est donne par :
donnees3.D.E = [105, 106, 104, 103] ;
donnees3.D.f = d9.D.f(r(1, :)) ;
donnees3.D.f(5) = 4 ;

%Traces de U et des projections :
%Ordre comme [Uc{1},Uc{2},Uc{3},Uc{4},Uc{5}] :
nfig.n = 6 ;
trac_solEc(donnees, Uc, 0, 1 :5, 6, nfig, [], X, Y, T) ;
figure(6) ;
title('Second espace de projection : fonction a projeter', ...
    'Position', [-0.5,-0.65,0]) ;
%Ordre comme [Uc{1},Uc{3},Uc{2},Uc{4},Uc{4}] :
nfig.n = 7 ;
trac_solEc(d9, Vb9, 0, 1 :5, 6, nfig, [], X, Y, T) ;
figure(7) ;
title(['Second espace de projection ordre :', ...
    ' u1, u3, u2, u4, u4'], 'Position', [-0.5,-0.7,0]) ;
%Ordre comme [Uc{1},Uc{2},Uc{3},Uc{4},Uc{5}] :
```

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/proj_deriv_EF3

```
nfig.n = 8 ;
trac_solEc(donnees3, Vc9, 0, 1 :5, 6, nfig, [], X, Y, T) ;
figure(8)
title(['Second espace de projection ordre : u1, u2, u3,', ...
      ' u4, u5 (non projete)'], ...
      'Position', [-0.5,-0.65,0]) ;

warning on ;
```

Fonction :

```
function [p, q, qx, qy, M] = interp_2D(x, y, z) ;
```

Interpolation polynômiale 2D. La commande

```
[P, Q, QX, QY, M] = interp_2D(X, Y, Z) ;
```

calcule le polynôme d'interpolation $p(x, y)$ de 2 variables tel que $p(X, Y) = Z$. Le nombre de composantes de X , Y et Z doit être identique et prendre une des valeurs suivantes : 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, *etc.* de la forme $n(n+1)/2$, avec n entier positif. Le degré d (*cf.* ci-après) du polynôme correspondant est alors donné par 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 *etc.* La valeur de d est mise dans M .

Pour avoir existence et unicité de p , X et Y doivent satisfaire des critères de non alignement comme satisfont par exemple les points donnés par la fonction `maillage_KR` suivante de maillage de triangle.

Le degré de p est ici défini comme étant la plus grande des valeur $i + j$ parmi les monômes $x^i y^j$ qui composent p .

La variable P , vecteur ligne, et Q matrice carrée, sont deux représentations de p et Qx et Qy représentent les dérivées $p_{,x}$ et $p_{,y}$ avec le même type de représentation que Q .

Ainsi P représente les coefficients de p dans la base des puissances croissantes de x puis de yx , à y fixé, puis de y^2x , *etc i.e.* la base :

$$1, x, x^2, \dots, x^M, y, xy, x^2y, \dots, x^{M-1}y, \dots, y^{M-1}, xy^{M-1}, y^M.$$

en posant $M = M$.

Ensuite Q représente p de la manière suivante : Q est une matrice carrée de taille $M+1$ telle que $Q(i, j)$ est le coefficient du monôme $x^{i-1}y^{j-1}$ de p si $2 \leq i+j \leq M+2$ et vaut 0 si $i+j > M+2$. En particulier en un point (x, y) , la valeur de p est donnée par

$$p(x, y) = \text{sum}(\text{sum}(Q \cdot (x.^{[0:M]} \cdot y.^{[0:M]})))$$

avec $(x, y) = (x, y) \in \mathbb{R}^2$.

Enfin Qx et Qy sont des matrices carrées de taille $M+1$ qui représentent, suivant le même principe que Q , les polynômes dérivés $p_{,x}$ et $p_{,y}$.

Variables d'entrée :

X, Y : `DOUBLE ARRAY`. Vecteurs ligne ou colonne. Le couple $(X(i), Y(i))$ représente les coordonnées du i -ème point d'interpolation.

Z : `DOUBLE ARRAY`. Vecteur ligne ou colonne. $Z(i)$ est la valeur en $(X(i), Y(i))$ de la fonction à interpoler.

Variables de sortie :

P : DOUBLE ARRAY. Vecteur ligne. Coefficients du polynôme d'interpolation dans la base suivante :

$$1, x, x^2, \dots, x^M, y, xy, x^2y, \dots, x^{M-1}y, \dots, y^{M-1}, xy^{M-1}, y^M.$$

avec $M = M$.

Q : DOUBLE ARRAY Matrice carrée de taille $M + 1$. Coefficients du polynôme d'interpolation. $Q(i, j)$ est le coefficient du monôme $x^{i-1}y^{j-1}$ de p si $2 \leq i+j \leq M+2$ et vaut 0 si $i+j > M+2$. En particulier en un point (x, y) , la valeur de p est donnée par

$$p(x, y) = \text{sum}(\text{sum}(Q.*(x.^[0:M]')*y.^[0:M])) ,$$

avec $(x, y) = (x, y)$.

Qx, **Qy** : DOUBLE ARRAY. Matrice carrée de taille $M + 1$. Représentent respectivement la dérivée par rapport à x et y du polynôme d'interpolation avec le même type de représentation que **Q**.

M : DOUBLE ARRAY. Degré du polynôme d'interpolation.

Remarque. Les polynômes représentés par **Qx** et **Qy** sont de degré $M - 1$, on les a cependant représenté avec des matrices de la même taille que **Q** qui représente, elle, un polynôme de degré M .

Exemples : Ces exemples sont donnés dans le `help` de la fonction.

Exemple 1 : Les points d'interpolation sont ici donnés par l'utilisateur. Ce sont les sommets et les points milieux d'arêtes du triangle de sommets $(-1, 0)$, $(1, 0)$ et $(0, 1)$ pour interpolation par un polynôme de degré au plus 2. On interpole ensuite la fonction $f(x, y) = 3 + 9x + x^2 + 6y + 10xy + 2y^2$. La fonction f étant un polynôme de degré 2, le calcul d'interpolation redonne donc f . L'exemple 1 du `help` de la fonction `interp_2D` est le suivant.

EXEMPLE 1

```
clear ;
x = [-1, 1, 0, 0, -1/2, 1/2] ;
y = [ 0, 0, 1, 0, 1/2, 1/2] ;
z = 3 + 9*x + x.^2 + 6*y + 10*x.*y + 2*y.^2 ;
[p, q, qx, qy, M] = interp_2D(x,y,z) ;
disp(p) ;
disp(q) ;
disp(qx) ;
disp(qy) ;
```

Ci-dessus on obtient :

$$\mathbf{p} = (3, 9, 1, 6, 10, 2), \mathbf{q} = \begin{pmatrix} 3 & 6 & 2 \\ 9 & 10 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

$$\mathbf{qx} = \begin{pmatrix} 9 & 10 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{qy} = \begin{pmatrix} 6 & 4 & 0 \\ 10 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Exemple 2 : On interpole la fonction f de l'exemple 1. Les points d'interpolation sont obtenus en partant d'un triangle K que l'on subdivise ensuite en M^2 sous-triangles égaux avec la fonction `maillage_KR` et $M = 2$ qui renvoie les coordonnées barycentriques des sommets du maillage obtenu (la valeur de ces coordonnées barycentriques est indépendante de K). La fonction `coord_maillage_KB` permet alors de passer des coordonnées barycentriques aux coordonnées cartésiennes. L'exemple 2 du `help` de la fonction `interp_2D` est le suivant.

EXEMPLE 2

```
clear ;
%Triangle
XT = [0, 1, 1/2] ;
YT = [0, 0, 1] ;
%Maillage du triangle
M = 2 ;
[T,N,L1,L2,L3] = maillage_KR(M) ;
[x,y] = coord_maillage_KB(L1, L2, L3, XT, YT) ;
z = 3 + 9*x + x.^2 + 6*y + 10*x.*y + 2*y.^2 ;
[p, q, qx, qy, M] = interp_2D(x,y,z) ;
disp(p) ;
disp(q) ;
disp(qx) ;
disp(qy) ;
```

On retrouve les résultats de l'exemple 1, les coordonnées cartésiennes obtenues sont

$$x = (0, 1/2, 1, 1/4, 3/4, 1/2), \quad y = (0, 0, 0, 1/2, 1/2, 1),$$

qui sont à nouveau les sommets et points milieux d'arête du triangle K .

Algorithme :

EF/trace_graphe_interp/interp_2D.m

On remet ici la partie déclarative de la fonction :

```
function [p, q, qx, qy, M] = interp_2D(x, y, z) ;
```

Soit $\ell = 1, 3, 6, 10, 15$ *etc.* une des valeurs prise par le nombre de termes de $\mathbf{x}$. Le degré du polynôme est alors donné par :

$$d = (-3 + \sqrt{1 + 8\ell})/2,$$

On note r_{ij} le coefficient du monôme $x^i y^j$ du polynôme d'interpolation inconnu noté p . Soit $x = \mathbf{X}(k)$, $y = \mathbf{Y}(k)$, $z = \mathbf{Z}(k)$, on a alors pour chaque valeur de k l'équation suivante (on omet la dépendance en k dans la notation de x , y et z) :

$$\begin{aligned} & r_{00} + r_{10}x + r_{20}x^2 + \cdots + r_{j0}x^j + \cdots + r_{d-1,0}x^{d-1} + r_{d0}x^d \\ & + r_{01}y + r_{11}xy + r_{21}x^2y + \cdots + r_{j1}x^jy + \cdots + r_{d-1,1}x^{d-1}y \\ & \quad \cdots \\ & + r_{0i}y^i + r_{1i}xy^i + r_{2i}x^2y^i + \cdots + r_{ji}x^jy^i + \cdots + r_{d-i,i}x^{d-i}y^i \\ & \quad \cdots \\ & + r_{0d}y^d = z. \end{aligned}$$

On pose comme vecteur inconnu $\mathbf{p}$ donné par :

$$\mathbf{p} = \begin{pmatrix} r_{00}, & r_{10}, & r_{20}, & \cdots, & r_{j0}, & \cdots, & \cdots, & r_{d-1,0}, & r_{d0}, \\ r_{01}, & r_{11}, & r_{21}, & \cdots, & r_{j1}, & \cdots, & \cdots, & r_{d-1,1}, & \\ \cdots & & & & & & & & \\ r_{0i}, & r_{1i}, & r_{2i}, & \cdots, & r_{ji}, & \cdots, & r_{d-i,i}, & & \\ \cdots & & & & & & & & \\ r_{0d} \end{pmatrix}.$$

Soit a la matrice dépendant de k définie par :

$$a_{ij} = x^i y^j, 0 \leq i, j \leq d,$$

alors les coefficients de la ligne k de la matrice A du système vérifié par $\mathbf{p}$ sont donnés par le vecteur :

$$A(k, :) = \begin{pmatrix} a_{00}, & a_{10}, & a_{20}, & \cdots, & a_{j0}, & \cdots, & \cdots, & a_{d-1,0}, & a_{d0}, \\ a_{01}, & a_{11}, & a_{21}, & \cdots, & a_{j1}, & \cdots, & \cdots, & a_{d-1,1}, & \\ \cdots & & & & & & & & \\ a_{0i}, & a_{1i}, & a_{2i}, & \cdots, & a_{ji}, & \cdots, & a_{d-i,i}, & & \\ \cdots & & & & & & & & \\ a_{0d} \end{pmatrix}.$$

La ligne $A(k, :)$ est donc formée successivement par les $d + 1$ premiers termes de la première colonne de a (*i.e.* la première colonne de a), puis par les d premiers termes de la colonne numéro 2 de a , *etc.*, puis des $d - i + 2$ premiers termes de la colonne numéro i de a *etc.* et du premier terme de la colonne numéro $d + 1$ (dernière colonne) de a .

On numérote les termes de a colonne par colonne, en posant b le vecteur :

$$\begin{pmatrix} 1, & 2, & \cdots & j, \cdots & d-1 & d, & d+1, \\ (d+1)+1, & (d+1)+2, & \cdots & (d+1)+j, \cdots & 2d, & 2d+1, \\ 2(d+1)+1, & 2(d+1)+2, & \cdots & 2(d+1)+j, \cdots & 3d+1, \\ \cdots & & & & & & \\ d(d+1)+1 \end{pmatrix}$$

on obtient

$$A(k, :) = a(b).$$

Cette relation est alors utilisée dans la fonction `interp_2D` pour la calcul de A .

On obtient $\mathbf{p}$ en résolvant le système obtenu. La matrice $\mathbf{q}$ qui représente le polynôme d'interpolation sous forme de matrice s'obtient alors de la manière suivante. Soit c le vecteur ligne nul de $(d + 1)^2$ termes. Si on pose

$$c(b) = \mathbf{p},$$

on obtient alors pour c :

$$c = \begin{pmatrix} \mathbf{p}(1), & \mathbf{p}(2) & , \cdots, & \mathbf{p}(d-1) & \mathbf{p}(d), & \mathbf{p}(d+1), \\ \mathbf{p}(d+2), & \mathbf{p}(d+3) & , \cdots, & \mathbf{p}(2d), & \mathbf{p}(2d+1), & 0, \\ \mathbf{p}(2d+2), & \mathbf{p}(2d+3) & , \cdots, & \mathbf{p}(3d), & 0, & 0, \\ \cdots & & & & & \\ \mathbf{p}(d(d+1)+1) & 0 & , \cdots, & 0, & 0, & 0, \end{pmatrix}$$

c'est à dire :

$$c = \begin{pmatrix} r_{00}, & r_{10}, & r_{20}, & \cdots, & r_{j0}, & \cdots, & r_{d-i,0}, & \cdots, & r_{d-1,0}, & r_{d0}, \\ r_{01}, & r_{11}, & r_{21}, & \cdots, & r_{j1}, & \cdots, & r_{d-i,1}, & \cdots, & r_{d-1,1}, & 0, \\ \cdots & & & & & & & & & \\ r_{0i}, & r_{1i}, & r_{2i}, & \cdots, & r_{ji}, & \cdots, & r_{d-i,i}, & \cdots, & 0, & 0, \\ \cdots & & & & & & & & & \\ r_{0d}, & 0, & 0, & \cdots, & 0, & \cdots, & 0, & \cdots, & 0, & 0. \end{pmatrix}$$

En mettant ce vecteur ligne sous forme de matrice carrée de taille $(d + 1)^2$, en remplissant les colonnes une à une, on obtient alors la matrice $\mathbf{q}$ en sortie de la fonction `interp_2D` avec $\mathbf{q}$ telle que $\mathbf{q}(i, j) = r_{i-1, j-1}$ pour $2 \leq i + j \leq d + 2$ (coefficient de $x^{i-1}y^{j-1}$ de p) et 0 sinon.

L'algorithme proprement dit est écrit de la manière suivante :

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/interp_2D`

- Calcul du degré du polynôme d'interpolation :

`L = length(x) ;`

`M = (-3 + sqrt(9 + 8*(L - 1)))/2 ;`

- Calcul de la matrice du système :

- Calcul du vecteur b ci-dessus

`b = find(rot90(tril(ones(M+1)),-1)) ;`

`b = b' ;`

Exemple pour $M = 3$ on a :

$$\text{ones}(M+1) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}, \quad \text{tril}(\text{ones}(M+1)) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

$$\text{rot90}(\text{tril}(\text{ones}(M+1)),-1) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$

$$\text{find}(\text{rot90}(\text{tril}(\text{ones}(M+1)),-1)) = (1, 2, 3, 4, 5, 6, 7, 9, 10, 13)',$$

et donc b est égal au vecteur ligne

$$b = (1, 2, 3, 4, 5, 6, 7, 9, 10, 13)$$

que nous disposons comme ci-dessus sous la forme

```

1,   2,   3,   4,
5,   6,   7,
9,  10,
13.
```

- Boucle sur les points à interpoler :

`for i=1 :length(x)`

`x1 = x(i).^[0 :M] ;`

`y1 = y(i).^[0 :M] ;`

`%Calcul de la matrice a`

`a = x1'*y1 ;`

`%Calcul de la ligne i du systeme`

`A(i, :) = a(b) ;`

`end`

- Fin de la boucle.

- Fin du calcul de la matrice du système.

Résolution du système d'interpolation pour le calcul de p qui est ensuite mis sous forme de vecteur ligne :

```
p = A\z( : ) ;
```

```
p = p' ;
```

- Calcul du vecteur c pour ensuite calculer q , c est d'abord initialisée à un vecteur ligne nul de taille $(d+1)^2$:

```
c((M+1)^2) = 0 ;
```

```
c(b) = p ;
```

- Calcul de q :

```
q = reshape(c,M+1,M+1) ;
```

- Calcul des dérivées :

```
for i=1 :L
```

```
    qx = repmat(0 :M,M+1,1)'.*q ;
```

```
    qy = repmat(0 :M,M+1,1).*q ;
```

```
end ;
```

Exemple, ci-dessus pour $M = 3$ on a :

$$\text{repmat}(0 : M, M + 1, 1) = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{pmatrix}$$

L'opération `repmat(0 :M,M+1,1)'.*q` multiplie le terme (i, j) de q , coefficient du monôme $x^{i-1}y^{j-1}$ du polynôme d'interpolation, par $i - 1$ et l'opération `repmat(0 :M,M+1,1).*q` multiplie ce terme par $j - 1$. Ceci correspond aux dérivations respectives par rapport à x et par rapport à y .

Ensuite on rajoute une ligne de 0 à qx et on supprime la première ligne de qx , ce qui abaisse d'une unité le degré en x du polynôme représenté par qx , *idem* sur les colonnes de qy pour la dérivation par rapport à y .

```
qx(end + 1, :) = zeros(1,M+1) ;
```

```
qx(1, :) = [] ;
```

```
qy(:, end + 1) = zeros(M+1,1) ;
```

```
qy(:, 1) = [] ;
```

Fonction :

```
function [TK, Nb, L1b, L2b, L3b] = maillage_KR(M) ;
```

Structure de maillage d'un triangle.

La commande

```
[TK, Nb, L1b, L2b, L3b] = maillage_KR(M) ;
```

subdivise un triangle de référence en M^2 sous-triangles égaux. La variable `TK` est la description des sommets du maillage, triangle par triangle, `Nb` les numéros de sommets qui ici vont de 1 à $(M+1)(M+2)/2$ et `L1b`, `L2`, `L3b` les coordonnées barycentriques de ces sommets. Les variables de sorties ne dépendent pas du choix du triangle (valeurs intrinsèques à M).

Ensuite pour mailler un triangle quelconque de sommets de coordonnées `XT`, `YT` en M^2 sous-triangles égaux, on utilisera la commande :

```
[X,Y] = coord_maillage_KB(L1b, L2b, L3b, XT, YT) ;
```

qui renvoie les coordonnées cartésiennes du maillage.

Variables d'entrée :

`M` : DOUBLE ARRAY. Variable de raffinement du triangle subdivisé en M^2 sous-triangles égaux. L'ordre de numérotation des triangles et des sommets est donné pour le cas $M = 4$ sur la figure 4.2-1 du Paragraphe 4.2.

Variables de sortie :

`TK` : DOUBLE ARRAY. Tableau à 3 colonnes. Description triangle par triangle des sommets du maillage du triangle. `TK(i,:)` désigne les numéros des trois sommets du triangle numéro i . Les sommets du triangle ont pour numéro 1, $M+1$ et $(M+1)(M+2)/2$.

`Nb` : DOUBLE ARRAY. Numéros des sommets du maillage. On a en sortie :

$$Nb = 1 : (M+1)(M+2)/2.$$

`L1b`, `L2b`, `L3b` : DOUBLE ARRAY. Vecteurs ligne. `L1b(i)` est la première coordonnée barycentrique du sommet numéro i du maillage. *Idem* pour `L2b` et `L3b` relativement à la deuxième et troisième coordonnée barycentrique. On a la relation

$$L1b(1) = L2b(M+1) = L3b((M+1)(M+2)/2) = 1.$$

Exemple : Cet exemple est tiré du `help` de la fonction `maillage_KR`, il donne la figure 4.2-1 du Paragraphe 4.2, avec $M = 4$.

EXEMPLE :

```
%Maillage du triangle de sommets de coordonnees x = (0.5,0,1),
```

```

%y = (1,0,0) en  $M^2 = 16$  sous-triangles.
close all ;
clear ;
M = 4 ;
[TK,Nb,L1b,L2b,L3b] = maillage_KR(M) ;
%Coordonnees du triangle a mailler :
XT = [0.5,0,1] ;
YT = [1,0,0] ;
%Coordonnees cartesiennes du triangle K maille :
[X,Y] = coord_maillage_KB(L1b, L2b, L3b, XT, YT) ;
figure(1) ;
%Trace des triangles du maillage :
for i=1 :size(TK,1), patch(X(TK(i, :)),Y(TK(i, :)),'w') ;
    hold on ;
    text(mean(X(TK(i, :))),mean(Y(TK(i, :))),num2str(i))
end ;
for i=1 :length(X)
    hold on
    text(X(i), Y(i), [' ', num2str(i)], 'Color','r')
end ;
plot(X,Y,'o') ;
%Numerotation des sommets des triangle :
text(XT(1) - 0.1, YT(1) + 0.05, ['sommet n\circ ', num2str(1)])
text(XT(2) - 0.08, YT(2) - 0.05, ['sommet n\circ ', num2str(2)])
text(XT(3) - 0.1, YT(3) - 0.05, ['sommet n\circ ', num2str(3)])
axis([-0.1,1.1,-0.1,1.1]) ;
title(['M = 4, Numerotation de triangles et des sommets du',...
        ' sous-maillage']) ;

```


Algorithme :

EF/trace_graphe_interp/maillage_KR.m

On remet ici la partie déclarative de la fonction :

```
function [TK, Nb, L1b, L2b, L3b] = maillage_KR(M) ;
```

Nous décrivons d'abord les variables A , $L1$, $L2$, $L3$, T suivantes qui sont utilisées dans l'algorithme. Cette description est faite dans le cas particulier de l'exemple donné dans le `help` de la fonction qui raffine le triangle avec $M = 4$ et dont le tracé est donné à la figure 4.2-1 du Paragraphe 4.2.

On part du triangle de sommets A_1 , A_2 , A_3 du schéma ci-dessous. On numérote 1, $M + 1 = 5$ et $(M + 1)^2 = 25$ ces trois sommets A_1 , A_2 et A_3 , puis on maille le carré obtenu par symétrie du triangle autour de l'axe A_1 , A_3 en M^2 sous-carrés :

↓ A_1 : Sommet 1

```

1-----6-----11-----16-----21
|         |         |         |         |
|   1   |   2   |   3   |   4   |
|         |         |         |         |
2-----7-----12-----17-----22
|         |         |         |         |
|   5   |   6   |   7   |   8   |
|         |         |         |         |
3-----8-----13-----18-----23
|         |         |         |         |
|   9   |  10  |  11  |  12  |
|         |         |         |         |
4-----9-----14-----19-----24
|         |         |         |         |
|  13   |  14   |  15   |  16   |
|         |         |         |         |
5-----10-----15-----20-----25
```

↑ A_2 : Sommet 2

↑ A_3 : Sommet 3

Les sommets obtenus sont numérotés, par colonnes, de 1 à 25 et les sous-carrés de 1 à 16.

La matrice A , dont nous donnons le principe de construction ci-après, est alors

définie sur cet exemple par

$$A = \begin{pmatrix} 1 & 6 & 6 & 11 & 11 & 16 & 16 & 21 \\ 2 & 7 & 7 & 12 & 12 & 17 & 17 & 22 \\ 2 & 7 & 7 & 12 & 12 & 17 & 17 & 22 \\ 3 & 8 & 8 & 13 & 13 & 18 & 18 & 23 \\ 3 & 8 & 8 & 13 & 13 & 18 & 18 & 23 \\ 4 & 9 & 9 & 14 & 14 & 19 & 19 & 24 \\ 4 & 9 & 9 & 14 & 14 & 19 & 19 & 24 \\ 5 & 10 & 10 & 15 & 15 & 20 & 20 & 25 \end{pmatrix}.$$

La matrice **A** contient alors les numéros de sommets suivants :

- ceux du carré numéro 1 :

$$A = \begin{pmatrix} 1 & 6 & \dots \\ 2 & 7 & \dots \\ & \dots & \end{pmatrix};$$

- puis du carré numéro 2 :

$$A = \begin{pmatrix} 1 & 6 & 6 & 11 & \dots \\ 2 & 7 & 7 & 12 & \dots \\ & \dots & \end{pmatrix};$$

- puis du carré numéro 3 :

$$A = \begin{pmatrix} 1 & 6 & 6 & 11 & 11 & 16 & \dots \\ 2 & 7 & 7 & 12 & 12 & 17 & \dots \\ & \dots & \end{pmatrix};$$

- puis du carré numéro 4 :

$$A = \begin{pmatrix} 1 & 6 & 6 & 11 & 11 & 16 & 21 \\ 2 & 7 & 7 & 12 & 12 & 17 & 22 \\ & \dots & \end{pmatrix};$$

Ensuite on passe à la seconde ligne de carrés, le carré numéro 5 a ses sommets donné aux lignes 3 et 4 de **A** par :

$$A = \begin{pmatrix} 1 & 6 & 6 & 11 & 11 & 16 & 21 \\ 2 & 7 & 7 & 12 & 12 & 17 & 22 \\ 2 & 7 & \dots & & & & \\ 3 & 8 & \dots & & & & \\ \dots & \dots & & & & & \end{pmatrix},$$

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/maillage_KR`

etc.

Ensuite $L1(i, j)$, $L2(i, j)$, $L3(i, j)$ sont définies comme les coordonnées barycentriques du sommet $A(i, j)$, ce qui donne sur cet exemple

$$L1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 \\ 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

etc.

On définit alors le tableau T par

$$T = \begin{pmatrix} 1 & 2 & 7 \\ 2 & 3 & 8 \\ 2 & 7 & 8 \\ 7 & 8 & 13 \\ 3 & 4 & 9 \\ 3 & 8 & 9 \\ 8 & 9 & 14 \\ 8 & 13 & 14 \\ 13 & 14 & 19 \\ 4 & 5 & 10 \\ 4 & 9 & 10 \\ 9 & 10 & 15 \\ 9 & 14 & 15 \\ 14 & 15 & 20 \\ 14 & 19 & 20 \\ 19 & 20 & 25 \end{pmatrix},$$

qui s'obtient en prenant :

- dans la ligne 1 du carré maillé, le triangle de sommets dans l'ordre local 1, 2, 7 du carré numéro 1, ce qui donne

$$T = \begin{pmatrix} 1 & 2 & 7 \\ & \dots & \end{pmatrix},$$

- dans la ligne 2 du carré maillé :

- le triangle de sommets dans l'ordre local 2, 3, 8 du carré 5,
- le triangle de sommets dans l'ordre local 2, 7, 8 du carré 5,
- le triangle de sommets dans l'ordre local 7, 8, 13 du carré 6,

ce qui donne

$$T = \begin{pmatrix} 1 & 2 & 7 \\ 2 & 3 & 8 \\ 2 & 7 & 8 \\ 7 & 8 & 13 \\ & \dots & \end{pmatrix},$$

- dans la ligne 3 du carré maillé :
- le triangle de sommets dans l'ordre local 3, 4, 9 du carré 9,
- le triangle de sommets dans l'ordre local 3, 8, 9 du carré 9,
- le triangle de sommets dans l'ordre local 8, 9, 14 du carré 10,
- le triangle de sommets dans l'ordre local 8, 13, 14 du carré 10,
- le triangle de sommets dans l'ordre local 13, 14, 19 du carré 11, ce qui donne :

$$T = \begin{pmatrix} 1 & 2 & 7 \\ 2 & 3 & 8 \\ 2 & 7 & 8 \\ 7 & 8 & 13 \\ 3 & 4 & 9 \\ 3 & 8 & 9 \\ 8 & 9 & 14 \\ 8 & 13 & 14 \\ 13 & 14 & 19 \\ & \dots & \end{pmatrix},$$

puis, de la même manière, en prenant les triangles de la dernière ligne du carré maillé.

L'algorithme est alors le suivant.

- Initialisation de A ci-dessus :

```
for i=1 :M
    A(2*i-1,1) = i ;
    A(2*i,1) = i+1 ;
end ;
for i=1 :M-1
    A( :,2*i) = A( :,2*i-1) + M + 1 ;
    A( :,2*i+1) = A( :,2*i) ;
end ;
i = M ;
A( :,2*i) = A( :,2*i-1) + M + 1 ;
```

- Initialisation des coordonnées barycentriques L1, L2, L3 ci-dessus :

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/maillage_KR`

```
i = 1 :M ;
```

```
J = 1 :M ;
```

- Le terme J ci-dessous est la matrice de 2M colonnes et de M lignes dont la ligne i vaut i :

```
J = J(ones(2*M,1), :) ;
```

- Le terme J ci-dessous est la matrice de 2M colonnes et de M lignes dont la ligne i vaut $(M - i + 1)/M$:

```
J = flipud(J)/M ;
```

Sur l'exemple cela donne :

$$J = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 \end{pmatrix}$$

dont les lignes composent la matrice de coordonnées barycentriques L1 donnée ci-dessus ainsi que L2 et L3, avec :

```
L1(2*i - 1,1 :2*M) = J ;
```

```
L1(2*i,1 :2*M) = J - 1/M ;
```

```
L3(1 :2*M,2*i - 1) = J' ;
```

```
L3(1 :2*M,2*i) = J' - 1/M ;
```

```
L3 = fliplr(L3) ;
```

```
L2 = 1 - L3 - L1 ;
```

• Construction de T :

```
k = 1 ;
```

```
for i=1 :M
```

```
    for j=1 :i
```

```
        MT = A(2*i-1 :2*i,2*j-1 :2*j) ;
```

```
        T(k, :) = [MT(1,1), MT(2,1), MT(2,2)] ;
```

```
        k = k+1 ;
```

```
    if j<i
```

```
        T(k, :) = [MT(1,1), MT(1,2), MT(2,2)] ;
```

```
        k = k+1 ;
```

```
    end ;
```

```
end ;
```

```
end ;
```

Cette boucle donne alors la valeur de T.

Dans l'exemple, la variable `MT` prend successivement dans cette boucle en i sur les lignes du carré maillé, les valeurs des numéros des quatres sommets du carré 1 puis, à la ligne 2 du maillage, des carrés 5, 6, puis, ligne 3, des carrés 9, 10, 11 et enfin 13, 14, 15 et 16 sur la ligne 4.

Sur la ligne i de carrés, on parcourt les carrés avec la variable j allant de 1 à i , sur le carré courant j de sommets décrit par `MT`, le terme `T(k, :)` prend les valeurs des sommets du triangle inférieur, k augmente alors d'une unité, ensuite si $j < i$, c'est à dire pour un carré entièrement dans le triangle maillé, `T(k, :)`, les valeurs des sommets du triangle supérieur.

- Valeur des coordonnées barycentriques des sommets du maillage du carré : `L1a` est le vecteur ligne de M^2 termes tel que `L1a(i)` est la coordonnée barycentrique du sommet numéro i du carré :

`L1a(A) = L1 ;`

`L2a(A) = L2 ;`

`L3a(A) = L3 ;`

- Ensemble des numéros des sommets de `T` :

`MM = sort(union(T( :),T( :))) ;`

Dans l'exemple on a :

$$MM = (1, 2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 15, 19, 20, 25).$$

Les sommet numéro `MM(i)` du maillage aura alors pour numéro i .

On donne maintenant le calcul des 4 variables de sortie.

- Coordonnées barycentriques des sommets du maillage du triangle. On pose `L1b(i)` est la coordonnée du sommet numéro `MM(i)` qui sera renuméroté ci-dessous par i . Donc `L1b(i)` sera alors la coordonnée barycentrique du sommet numéro i :

`L1b = L1a(MM) ;`

`L2b = L2a(MM) ;`

`L3b = L3a(MM) ;`

Le sommet numéro `MM(i)` du maillage est renuméroté ci-dessous par i . Les numéros de `T` sont dans l'intervalle compris entre 1 et M^2 car on a choisi une numérotation des sommets des carrés. On renumérote les sommets du triangle maillé avec des numéros allant de 1 à $(M+1)(M+2)/2$, le résultat est mis dans la variable `TK` de sortie.

Sur l'exemple, les numéros intervenant dans `T` sont ceux donnés par `MM` :

$$1, 2, 3, 4, 5, 7, 8, 9, 10, 13, 14, 15, 19, 20, 25,$$

ils sont remplacés respectivement par

$$1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15.$$

4.3. DESCRIPTION DES FONCTIONS FONCTION `trace_graphe_interp/maillage_KR`

- Construction du tableau des numéros de sommets, les sommets de `T` sont renumérotés de 1 à $\text{length}(\text{MM}) = (M + 1)(M + 2)/2$:

```
TK = zeros(size(T)) ;  
for i=1 :length(MM) ;  
    I = find(T==MM(i)) ;  
    TK(I) = i ;  
end ;
```

- Numéros de sommets :

```
Nb = 1 :max(max(TK)) ;
```

Fin de l'algorithme.

Fonction :

```
function [X, Y] = coord_maillage_KB(L1, L2, L3, M1, M2) ;
```

Calcul de coordonnées de points à partir de leurs coordonnées barycentriques.

La commande

```
[X,Y] = coord_maillage_KB(L1, L2, L3, M1, M2) ;
```

calcule les coordonnées cartésiennes $X(i)$ et $Y(i)$ du point de coordonnées barycentriques $L1(i)$, $L2(i)$, $L3(i)$ relatives au triangle de sommets de coordonnées représentées par les vecteurs lignes ou colonnes de trois termes $M1$ et $M2$.

Variables d'entrée :

$L1$, $L2$, $L3$: DOUBLE ARRAY. Vecteurs lignes ou bien colonnes. Les termes $L1(i)$, $L2(i)$ et $L3(i)$ sont les coordonnées barycentriques relatives au triangle de sommets $M1$ et $M2$, dont on calcule les coordonnées cartésiennes.

$M1$, $M2$: DOUBLE ARRAY. Vecteur ligne ou colonne. Le couple $(M1(i), M2(i))$ est la coordonnée du sommet numéro i du triangle utilisé pour les coordonnées barycentriques.

Variables de sortie :

X , Y : DOUBLE ARRAY. Vecteur ligne. Le couple $(X(i), Y(i))$ est la coordonnée du point de coordonnées barycentriques $L1(i)$, $L2(i)$, $L3(i)$ relativement au triangle de coordonnées $M1$ et $M2$.

Exemple : Exemple d'utilisation de la fonction sur un triangle raffiné par `maillage_KR`. Cet exemple est tiré du `help` de la fonction `coord_maillage_KB` et illustre la numérotation des sous-triangles du maillage du triangle réalisé par `maillage_KR`.

EXEMPLE

```
%Coordonnees du triangle :
close all ;
clear ;
M1 = [0.5,0,1] ;
M2 = [1,0,0] ;
M = 6 ;
%Raffinement du triangle :
[T,A,L1,L2,L3] = maillage_KR(M) ;
[X,Y] = coord_maillage_KB(L1, L2, L3, M1, M2) ;
%Trace du maillage :
```


4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/coord_maillage_KB

```
s = size(T) ;  
R = 'rbrb' ;  
for i=1 :s(1),  
    m = R(mod(i,4) + 1) ;  
    patch(X(T(i, :)), Y(T(i, :)), m), hold on ;  
    text(mean(X(T(i, :))), mean(Y(T(i, :))), num2str(i)) ;  
end ;
```

Algorithme :

Il s'agit des deux commandes suivantes qui utilisent la définition des coordonnées barycentriques.

```
X = M1(1)*L1( :)' + M1(2)*L2( :)' + M1(3)*L3( :)' ;
```

```
Y = M2(1)*L1( :)' + M2(2)*L2( :)' + M2(3)*L3( :)' ;
```

Fonction :

```
function [Xc, Yc, Tc, varargout] = ddl(donnees, varargin) ;
```

Calcul des degrés de liberté *ddl* de chaque type d'espace d'EF qui intervient dans la composition de l'espace EF produit *E*.

La commande

```
[Xc, Yc, Tc] = ddl(donnees) ;
```

calcule les variables *Xc*, *Yc* et *Tc* qui décrivent les *ddl* de chaque type d'espace d'EF qui intervient dans la composition de l'espace EF produit *E* et dont la liste est donnée par *donnees.D.E*. Le maillage utilisé est celui stocké dans le fichier de nom donné par *donnees.nom_mail* et qui contient les variables *Noeu*, *X*, *Y*, *T*, *Front*. Le résultat ne dépend pas des valeurs des variables *Noeu* et *Front*. La variable *Noeu* peut ne pas être initialisée et la variable *Front* mise à [].

La commande

```
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []);
```

ou la commande

```
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, Front);
```

donnent le même résultat sans utiliser le fichier de maillage.

Les commandes du type

```
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, ...);
```

permettent de plus de récupérer les variables optionnelles *Ed* et *Arc* telles que *Ed* donne la dimension de chaque type d'espace d'EF choisi et telles que *Arc* décrive, pour chaque type d'espace d'EF, les *ddl* situés sur les arêtes du maillage.

Remarque. Dans sa version actuelle la fonction *ddl* ne dépend pas de *Front* qui peut valoir [].

Variables d'entrée :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF.

Pour les appels du type

```
ddl(donnees) ;
```

les champs à initialiser sont

donnees.D.E,

donnees.nom_mail.

Le résultat ne dépend pas des valeurs des variables *Noeu* et *Front* du fichier de maillage, *Noeu* peut ne pas être initialisée, la variable *Front* peut être mise à [].

Pour les appels du type :

```
ddl(donnees, X, Y, T, []);
```

ou

```
ddl(donnees, X, Y, T, Front) ;
```

avec appel avec des variables de maillage `X`, `Y`, `T`, l'unique champ à initialiser est `donnees.D.E.`

Variables d'entrées optionnelles :

`X,Y` : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`T` : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

`Xc`, `Yc` : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs `Xc{i}` et `Yc{i}` sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

`Tc` : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie optionnelles :

`Ed` : DOUBLE ARRAY. Vecteur ligne. `Ed(i)` est la dimension du i -ème type d'espace d'EF choisi.

`Arc` : CELL ARRAY. Vecteur ligne. `Arc{i}` est une matrice de type DOUBLE ARRAY décrivant les *ddl* arêtes par arêtes du i -ème type d'espace d'EF choisi, *cf.* le Tableau 2.2.3-1b du Paragraphe 2.2.3, `Arc{i}` est un vecteur colonne nul dans le cas d'un espace d'EF qui n'a pas de *ddl* d'arêtes comme les espaces d'EF discontinus de numéros 100 à 199 et de numéro 300. La valeur de `Arc{i}` est donnée par la valeur de la variable `Ad` calculée dans `maillage_ddl` dans le cas du i -ème type d'espace d'EF choisi.

Exemple : Cet exemple est donné dans le `help` de la fonction `ddl`. La construction et l'utilisation de `Front` est facultative. Cet exemple calcule et trace les *ddl* de 5 types d'espaces d'EF donné par le maillage du carré unité par 8 triangles de la figure 4.3-5 et qui sont dans l'ordre les espaces d'EF :

$$P_1^c, P_2^c, P_5^c, P_3^d, P_2^c.$$

L'exemple est alors le suivant :

EXEMPLE

```
clear ;
```

```

close all ;
%Maillage :
T = [1,5,4 ; 1,2,5 ; 2,6,5 ; 2,3,6 ; 4,8,7 ;
      4,5,8 ; 5,9,8 ; 5,6,9] ;
X = [0, 0.5, 1, 0, 0.5, 1, 0, 0.5, 1] ;
Y = [0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1] ;
Front{1,1} = [7,8,9] ;
Front{2,1} = [9,6,3] ;
Front{3,1} = [3,2,1] ;
Front{4,1} = [1,4,7] ;
%Types d'espaces EF (le meme espace peut figurer
%  plusieurs fois) :
donnees.D.E = [1,2,5,103,2] ;
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, X, Y, T, Front) ;
%Trace des ddl de chaque espace :
for k=1 :length(donnees.D.E)
subplot(2,3,k) ;
hold on
    for i=1 :size(T,1), patch(X(T(i, :)), Y(T(i, :)), 'w'), end ;
    title(['espace numero ', num2str(k)]) ;
    plot(Xc{k}, Yc{k}, 'o') ;
end ;

```

Algorithme :

EF/trace_graphe_interp/ddl.m

On remet ici la partie déclarative de la fonction :

```
function [Xc, Yc, Tc, varargin] = ddl(donnees, varargin) ;
```

- Initialisations diverses :

- S'il n'y a qu'une variable d'entrée (pas de variables optionnelles), chargement du maillage :

```
load(donnees.nom_mail, 'X', 'Y', 'T', 'Front') ;
```

- Variable décrivant les espaces d'EF :

```
D = donnees.D ;
```

- Numérotation des arêtes et description par arêtes du maillage :

On calcule les variables suivantes.

Am : Tableau qui décrit arête par arête les numéros des sommets du maillage,

Pa : Tableau de correspondance entre chaque numéro d'arête et :

- le numéro du triangle auquel elle appartient,
- son numéro local sur le triangle,
- *idem* pour l'éventuel second triangle,
- et enfin le numéro de morceau de frontière qui porte l'arête (0 pour les arêtes intérieures ou si **Front** []).

Ensuite les arêtes ont une numérotation locale sur chaque morceau de frontière. Les variables **Pfc** et **Pf** établissent une la correspondance entre le numéro local d'une arête d'un morceau de frontière *j* et :

- le numéro du triangle auquel elle appartient,
- son numéro local sur le triangle,
- son numéro global d'arête,

de la manière suivante.

Pfc{*j*}(*i*, :) est un vecteur ligne de 3 termes qui donne la correspondance pour l'arête *i* de la frontière numéro *j*.

Pf(*i*, :) est un vecteur ligne de 4 dont les 3 premiers termes donnent la correspondance pour l'arête *i* de la frontière numéro **Pf**(*i*, 4).

Ces deux variables contiennent la même information. Si **Front** = [] alors ces deux variables valent [] aussi.

Pour la description des variables **Am**, **Pa**, **Pfc** et **Pf** on pourra aussi se référer à la description de la fonction `maillage_triangle_arete_ps`

Le calcul est effectué par :

```
[Am, Pa, Pfc, Pf] = maillage_triangle_arete_ps(T, X, Y, Front) ;
```

- Calcul des degrés de libertés pour chaque type d'espace d'EF. Les variables de sortie calculées **Xc**, **Yc**, **Tc**, **Arc** et **Ed** sont celles décrites au Tableau 2.2.3-1b) du Paragraphe 2.2.3.

- Boucle en *i* sur les types d'espaces d'EF choisis

```
for i=1 :length(D.E)
```

```
    %Td : DDL triangles par triangles de l'espace numero D.E(i),
```

```
    %X2, Y2 : coordonnees des DDL de l'espace numero D.E(i),
```

```
    %Ad : DDL aretes par aretes (pour les cb) :
```

```
    [Td, Ad, X2, Y2] = maillage_ddl(D.E(i), T, Am, Pa, X, Y) ;
```

```
    %Mise des resultats relatifs a l'espace D.E(i) dans des
```

```
    %les cellules i de Tc, Xc, Yc, Arc
```

```
%DDL
Tc{i} = Td ;
%Coordonnees des DDL
Xc{i} = X2 ;
Yc{i} = Y2 ;
%DDL par aretes (variable optionnelle)
Arc{i} = Ad ;
%Nombre de degres liberte de l'espace (variable optionnelle)
Ed(i) = length(Xc{i}) ;
end ;
```

- Fin de la boucle en i
- Initialisation des variables optionnelles de sortie :

```
varargout{1} = Ed ;
varargout{2} = Arc ;
```

Fonction :

```
function Z = pt_tri(Ac, Bc, T, X, Y, varargin) ;
```

Correspondance entre les points d'un ensemble fini de $\mathbb{R}^2$ donné et leur (ou leurs) éventuel triangle d'appartenance.

La commande

```
Z = pt_tri(Ac,Bc,T,X,Y) ;
```

renvoie la variable **Z** tel que **Z(i, j)** vaut 1 si le point de coordonnées (**Ac(j)**, **Bc(j)**) appartient au triangle numéro *i* du maillage défini par les variables **X**, **Y**, **T** et vaut 0 sinon. La liste des numéros de triangles auquel appartient le point numéro *i* est donnée par le vecteur **find(Z(:, i) == 1)**. Cette liste peut être vide et valoir alors []. Lors de l'exécution de la fonction **pt_tri**, un **warning** signale chaque point qui n'appartient à aucun triangle.

La commande

```
Z = pt_tri(Ac,Bc,T,X,Y,tol) ;
```

effectue la même opération et introduit une tolérance pour l'appartenance d'un point à un triangle. Ceci permet de tenir compte du fait qu'un point puisse appartenir à un triangle mais qu'il ne soit pas détecté comme tel par la fonction **pt_tri** à cause des erreurs d'arrondis (dans ce cas un **warning** s'affiche). On utilisera alors la variable **tol** pour prendre en compte ce point.

Variables d'entrée :

Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en *x* et en *y* des points dont on cherche les triangles d'appartenance.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

X, Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables d'entrée optionnelle

tol : DOUBLE ARRAY. Scalaire qui indique une tolérance pour l'appartenance d'un point à un triangle. Il peut arriver qu'un point appartienne à un triangle, mais que la fonction **pt_tri** ne le signale pas en sortie à cause de l'erreur d'arrondi. On posera alors **tol** de l'ordre de l'erreur machine **eps** ou sinon à zéro.

Variables de sortie :

Z : DOUBLE ARRAY. Le nombre de lignes de Z est égal au nombre de lignes de T *i.e.* le nombre de triangles de maillages et le nombre de colonnes à la longueur de Ac (nombre de points à examiner). On a $Z(i, j) = 1$ si le point de coordonnées cartésiennes $(Ac(j), Bc(j))$ appartient au triangle numéro i et vaut 0 sinon.

Exemple : Cet exemple est tiré du `help` de la fonction `pt_tri`. On détermine lequel des deux triangles du maillage du carré unité de la figure 4.3-7 appartiennent les points de coordonnées $(x_i, y_i) = ((i - 1)/10, 0.5)$, avec $1 \leq i \leq 11$.

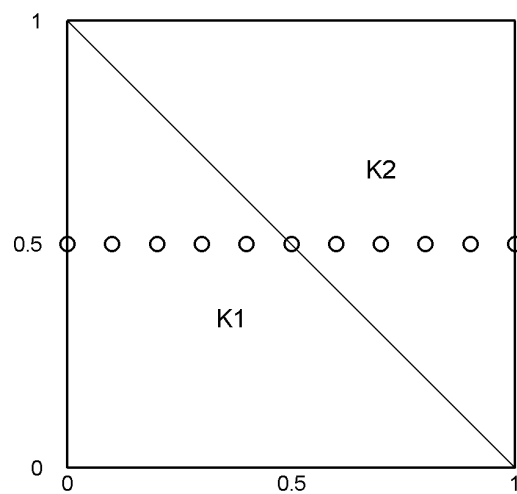


Figure 4.3-7. Détermination de l'appartenance au triangle numéro 1 ou 2 des points marqués par $\circ$.

EXEMPLE sur deux triangles :

```
T = [1,2,3;2,3,4] ;
X = [0,1,0,1] ;
Y = [0,0,1,1] ;
Ac = 0 :0.1 :1 ;
Bc = 0.5*ones(1,11) ;
Z = pt_tri(Ac, Bc, T, X, Y) ;
```

La valeur de Z trouvée est alors de

$$Z = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

Par exemple l'instruction `find(Z(:,6) == 1)` renvoie la valeur $(1,2)'$ qui montre que le point au milieu du carré appartient aux deux triangles.

Algorithme :

EF/trace_graphe_interp/pt_tri.m

On remet ici la partie déclarative de la fonction :

```
function Z = pt_tri(Ac, Bc, T, X, Y, varargin) ;
```

- Préallocation mémoire de la variable de sortie Z :

```
Z = sparse(size(T,1), length(Ac)) ;
```

- Initialisation de tol, tolérance pour l'appartenance à un triangle.

```
if isempty(varargin)
```

```
    tol = 0 ;
```

```
else
```

```
    tol = varargin{1} ;
```

```
end ;
```

- On calcule, pour chaque triangle, les coordonnées barycentriques relatives au triangle courant des points de coordonnées cartésiennes données par les vecteurs lignes Ac, Bc.

a) Calcul du second membre du système vérifié par les coordonnées barycentriques, ce second membre est identique pour tous les triangles :

```
BMAT = [Ac ; Bc ; ones(1, length(Ac))] ;
```

b) Pour chaque triangle, résolution du système des coordonnées barycentriques pour les points de coordonnées cartésiennes données par les vecteurs lignes Ac, Bc :

```
for i=1 :size(T,1)
```

```
    %Lignes 1 et 2 : coordonnees des sommets du triangle
```

```
    AMAT = [X(T(i,1 :3)) ; Y(T(i,1 :3)) ; 1,1,1] ;
```

```
    CMAT = AMAT\BMAT ;
```

- Si les trois coordonnées barycentriques d'un point sont comprises entre 0 et 1 alors ce point appartient au triangle courant :

```
    Z(i, :) = all((CMAT<=1 + tol).*(CMAT>=0 - tol)) ;
```

```
end ;
```

- Affichage d'un warning si un point n'appartient à aucun triangle :

```
for i=1 :length(Ac) ;
```

```
    numK = find(Z(:,i)==1) ;
```

```
    if isempty(numK)
```

```
        warning(['le point numero ', num2str(i), ...  
                ' n'appartient a aucun triangle']) ;
```

```
    end ;
```

```
end ;
```

Fonction :

```
function [U, varargout] = valeur_fonc_EF(donnees, Uc, der, Ac, ...
                                         Bc, numK, X, Y, T, Xc, Yc, Tc, r) ;
```

Valeur d'une fonction de l'espace EF produit E courant en des points d'un triangle donné.

La commande

```
U = valeur_fonc_EF(donnees, Uc, der, Ac, ...
                   Bc, numK, X, Y, T, Xc, Yc, Tc, R) ;
```

calcule les valeurs suivante.

Soit E l'espace EF produit représenté par les variables `donnees.D.E` et `donnees.D.f` et le maillage X, Y et T . Soit f la fonction de E représentée par la variable Uc , composantes de f dans la base de E choisie dans le code. Soit f_i la composante de f représentée par $Uc\{i\}$. Alors U est le vecteur de type `CELL ARRAY` de même taille que Uc et tel que $U\{i\}$ est le vecteur colonne des valeurs de $f_{i,der}(Ac, Bc)$, où Ac et Bc sont deux vecteurs lignes tels que les points de coordonnées cartésiennes $(Ac(j), Bc(j))$ appartiennent au triangle numéro `numK` du maillage et der la valeur donnée par $der = 0, 1, 2$. Le calcul est effectué pour i appartenant à r et sinon on a $U\{i\} = []$.

Lorsque $(Ac(j), Bc(j))$ n'est pas dans le triangle K de numéro `numK`, c'est un calcul d'extrapolation de $f_{i,der}(Ac(j), Bc(j))$ qui est réalisé : $f_{i,der}$ restreint à K est un polynôme p_K sur K , on prend alors la valeur du prolongement (unique) de p_K sur $\mathbb{R}^2$ au point $(Ac(j), Bc(j))$.

Les variables Xc et Yc ne sont pas utilisées dans cette version, elle peuvent être mises à `[]`. La variable `varargout` n'est pas non plus utilisée dans cette version.

Variables d'entrée :

donnees : `STRUCT ARRAY`. Variable décrivant les données du problème EF. La fonction dépend seulement des champs `donnees.D.E`, `donnees.D.f` qui, avec le maillage, définissent l'espace EF produit E . Les autres champs sont facultatifs.

Uc : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (cf. Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

der : `DOUBLE ARRAY`. Les valeurs de $der = 0, 1, 2$ indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

4.3. DESCRIPTION DES FONCTIONS F^N trace_graphe_interp/valeur_fonc_EF

Ac, Bc : DOUBLE ARRAY. Vecteurs lignes des coordonnées cartésiennes en x et en y des points pour lesquels on calcule la valeur de la fonction. Les points de coordonnées cartésiennes $(Ac(j), Bc(j))$ doivent appartenir au triangle numéro numK du maillage, sinon c'est un calcul d'extrapolation qui est effectué

numK : DOUBLE ARRAY. Scalaire, désigne le numéro de triangle auquel appartiennent les points de coordonnées cartésiennes **Ac** et **Bc** et sur lesquels on calcule les valeurs de la fonction représentée par **Uc**.

X, Y : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

T : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Xc, Yc : CELL ARRAY. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type DOUBLE ARRAY. Les vecteurs $Xc\{i\}$ et $Yc\{i\}$ sont les vecteurs des coordonnées en x et y des *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3. Non utilisées dans cette version, peuvent être mises à $\square$.

Tc : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice $Tc\{i\}$ décrit triangle par triangle les *ddl* du i -ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

r : DOUBLE ARRAY. Variable portant sur les composantes de l'espace EF produit E . Vecteur ligne ou colonne, $r(i)$ désigne la composante de la fonction dont on veut les valeurs.

Variables de sortie :

U : DOUBLE ARRAY. Le terme **U** est de même taille que la variable d'entrée **Uc**. Le terme $U\{i\}$ est le vecteur colonne des valeurs de $f_{i,der}(Ac, Bc)$, où **Ac** et **Bc** sont deux vecteurs lignes tels que les points de coordonnées cartésiennes $(Ac(j), Bc(j))$ appartiennent au triangle numéro numK du maillage. Le calcul est effectué pour i appartenant à **r** et sinon on a $U\{i\} = \square$. Lorsque $(Ac(j), Bc(j))$ n'est pas dans le triangle de numéro numK, c'est un calcul d'extrapolation de $f_{i,der}(Ac(j), Bc(j))$ qui est réalisé.

varargout : La variable **varargout** n'est pas utilisé dans cette version.

Exemple 1 : Cet exemple est tiré du **help** de la fonction ou est mis en œuvre par la commande

```
cd C :/EF/exemples/trace_graphe_interp, exemple_valeur_fonc_EF ;
```

Dans cet exemple on considère le maillage du carré unité en deux triangles de la figure 4.3-7 et l'espace EF produit $E = P_1^c \times P_3^c \times P_1^c$. On calcule les valeurs de la fonction f définie par $f(x, y) = (x, x^3 + y^3, y)$ en des points de chacun des deux triangles. Le fichier de commandes de cet exemple est le suivant.

EF/exemples/trace_graphe_interp/exemple_valeur_fonc_EF.m

```
%EXEMPLE 1 sur deux triangles :
clear ;
close all ;

%Maillage :
T = [1,2,3;2,3,4] ;
X = [0,1,0,1] ;
Y = [0,0,1,1] ;

%Espace EF :
donnees.D.E = [1,3] ;
donnees.D.f = [1,2,1] ;
[Xc, Yc, Tc] = ddl(donnees, X, Y, T, []) ;

%Definition d'une fonction de l'espace EF :
Uc{1} = Xc{1}' ;
Uc{2,1} = Xc{2}'.^3 + Yc{2}'.^3 ;
Uc{3,1} = Yc{1}' ;

%Valeurs sur le triangle 1 :
r = [1 :3] ;
der = 0 ;
numK = 1 ;
Ac = [0.2, 0.4, 0.6, 0.8] ;
Bc = [0.6, 0.5, 0.4, 0.3] ;
U1 = valeur_fonc_EF(donnees, Uc, der, Ac, ...
                    Bc, numK, X, Y, T, Xc, Yc, Tc, r) ;

%Valeurs sur le triangle 2 :
numK = 2 ;
Ac = [0.2, 0.4, 0.6, 0.8] ;
Bc = [0.9, 0.7, 0.6, 0.5] ;
U2 = valeur_fonc_EF(donnees, Uc, der, Ac, ...
                    Bc, numK, X, Y, T, Xc, Yc, Tc, r) ;
```

Exemple 2 : Cet exemple est le second exemple du `help` de la fonction. Dans cet exemple on considère à nouveau le maillage du carré unité en deux triangles de la figure 4.3-7 et l'espace EF produit E (restreint ici à un seul espace) $E = P_3^c$. On trace sur le carré de sommets $(-0.2, -0.2)$, $(1.2, -0.2)$ $(1.2, 1.2)$ $(-0.2, 1.2)$ l'unique polynôme de $P_3^c(\mathbb{R}^2)$ dont la restriction au triangle 1 est la fonction bulle de E qui vaut 0 au centre de gravité du triangle 1 et 0 aux autres *ddl* de E . Pour cela, dans l'exemple, `Uc{1}` représente cette fonction bulle de E . Les commandes de cet exemple sont les suivantes.

EXEMPLE 2 sur deux triangles :

```
clear ;
close all ;

%Maillage :
T = [1,2,3 ; 2,3,4] ;
X = [0,1,0,1] ;
Y = [0,0,1,1] ;

%Espace EF :
donnees.D.E = 3 ;
donnees.D.f = 1 ;
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, X, Y, T, []);

%Definition de Uc fonction bulle P3 sur le triangle 1 :
Uc{1} = zeros(Ed,1) ;
Uc{1}(Tc{1}(1,end)) = 1 ;

%Representation sur un rectangle de P polynome P3 dont la res-
% triction au triangle 1 est la fonction bulle (pour la
% representation graphique on tronque a -1.5 si P <=-1.5) :
%
numK = 1 ;
der = 0 ;
r = 1 ;
[Ac,Bc] = meshgrid(-0.2 :0.02 :1.2, -0.2 :0.02 :1.2) ;
U3 = valeur_fonc_EF(donnees, Uc, der, Ac( : )', ...
```

```

        Bc( :)', numK, X, Y, T, Xc, Yc, Tc, r) ;
Z = reshape(U3{r},size(Ac,1),size(Ac,2)) ;
I = find(Z <-1.5) ;
Z(I) = -1.5 ;
surf(Ac,Bc,Z) ;
set(gca,'CameraPosition',[0.93, -0.8, 34]) ;
hold on
patch(X(1 :3),Y(1 :3),'w','Linewidth',2) ;
title(['Polynome dont la restriction au triangle 1 est', ...
      ' la fonction bulle (on tronque au seuil inf. de -1.5)']) ;

```

Algorithme :

EF/trace_graphe_interp/valeur_fonc_EF.m

On remet ici la partie déclarative de la fonction :

```

function [U, varargout] = valeur_fonc_EF(donnees, Uc, der, Ac, ...
        Bc, numK, X, Y, T, Xc, Yc, Tc, r) ;

```

Cet algorithme reprend en partie l'algorithme de la fonction `interp_EF_ddl2`, où par rapport à `interp_EF_ddl`, on calcule les coordonnées barycentriques des *ddl* par résolution du système que vérifient les coordonnées barycentriques et où on utilise la fonction `F_base_T` à la place de la fonction `F_base`.

- Initialisations :

```
D = donnees.D ;
```

```
rep = D.E(D.f) ;
```

- Calcul des coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) des sommets du triangle numéro `numK` (pour le maillage EF) et de la variable $\mathbf{de} = (a_1, b_1, c_1) \cdot (b_2 - c_2, -a_2 + c_2, -b_2 + a_2)$ (l'opérateur $\cdot$ est ici le produit scalaire) à partir des coordonnées des sommets du triangle courant donnés par M1 et M2 ci-dessous :

```
M1 = X(T(numK,1 :3)) ;
```

```
M2 = Y(T(numK,1 :3)) ;
```

```
a1 = M1(1) ;
```

```
b1 = M1(2) ;
```

```
c1 = M1(3) ;
```

```
a2 = M2(1) ;
```

```
b2 = M2(2) ;
```

```
c2 = M2(3) ;
```

```
de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;
```

- Boucle sur les composantes de l'espace EF produit E :

```
for i=1 :length(rep)
    if any(i==r( :))
```

- Valeurs des composantes dans la base EF de la fonction représentée par $Uc\{i\}$:

```
    V = Uc{i}(Tc{D.f(i)}(numK, :)) ;
    d = der ;
    e = rep(i) ;
```

- 1) Comme dans `interp_EF_ddl2` dans le cas des *ddl*, on calcule les coordonnées barycentriques des points de coordonnées Ac et Bc par résolution du système que vérifient les coordonnées barycentriques :

```
    AMAT = [a1, b1, c1 ; a2, b2, c2 ; 1, 1, 1] ;
    BMAT = [Ac ; Bc ; ones(1,length(Ac))] ;
    CMAT = AMAT\BMAT ;
    L1b = CMAT(1, :) ;
    L2b = CMAT(2, :) ;
    L3b = CMAT(3, :) ;
```

- Valeur de la dérivée spécifiée par d de la fonction représentée par $Uc\{i\}$ aux points de coordonnées Ac et Bc à partir des coordonnées barycentriques de ces points, du numéro d'espace et des paramètres du triangle courant.
-

- 2) Comme dans `interp_EF_ddl2` on utilise la fonction `F_base_T` à la place de la fonction `F_base` utilisée dans `interp_EF_ddl`.

```
    [z, dim] = F_base_T(e, d, [], [], a1, a2, b1, b2, c1, ...
                        c2, de, D, L1b, L2b, L3b) ;

    U{i,1} = sum(diag(V)*z,1)' ;
else
    U{i,1} = [] ;
end ;
end ;
```

Fonction :

```
function [U, A, B] = trac_sol_coup(donnees, ...
```

```
Ac, Bc, tol, Uc, der, r, M, varargin);
```

Calcule en des points quelconques de $\mathbb{R}^2$ des valeurs des fonctions de l'espace EF produit E et trace des coupes. Permet aussi de tracer des champs de vecteurs. Le tracé est effectué en 3D.

La commande

```
[U, A, B] = trac_sol_coup(donnees, Ac, Bc, tol, Uc, der, r, M);
```

calcule les valeurs suivantes.

- Soit E l'espace EF produit représenté par les variables `donnees.D.E` et `donnees.D.f` et le maillage X, Y et T contenu dans le fichier de nom `donnees.nom_mail`.
- Soit A et B les coordonnées cartésiennes des points déduits de Ac, Bc et M de la manière suivante. Pour $M = 1$ on pose $A = Ac$ et $B = Bc$. Pour $M > 1$, chaque intervalle $[(Ac(i), Bc(i)), (Ac(i+1), Bc(i+1))]$ est subdivisé en M intervalles égaux, A et B sont alors les coordonnées cartésiennes de la réunion sur tous les intervalles $[(Ac(i), Bc(i)), (Ac(i+1), Bc(i+1))]$ de ces points.
- Soit f la fonction de l'espace EF produit E décrite par ses composantes Uc dans la base de E choisie dans le code.

Alors U est le vecteur de type `CELL` de même taille que Uc et tel que pour $der = 0, 1, 2$ et i dans l'ensemble décrit par le vecteur r , $U\{i\}$ est la valeur aux points de coordonnées A et B de $f_{i,der}$. On pose $U\{i\} = []$ pour les autres valeurs de i . On pourra poser $der = []$ à la place de $der = 0$.

Si un point appartient à plusieurs triangles la dérivée choisie en ce point est la moyenne des valeurs des dérivées en ce point de chaque restriction de f à chacun des triangles.

On peut utiliser la commande longue (*cf.* ci-après) :

```
[U, A, B] = trac_sol_coup(DONNEES, Ac, Bc, tol, Uc, ...
```

```
DER, R, M, NFIG, VEC, X, Y, T);
```

ou

```
[U, A, B] = trac_sol_coup(DONNEES, Ac, Bc, tol, Uc, ...
```

```
DER, R, M, NFIG, VEC, X, Y, T, Tc);
```

avec $NFIG = []$, $VEC = []$. Dans les deux cas les variables de maillage X, Y, T et la variable décrivant les numéros de *ddl* par espaces et par triangles Tc sont utilisées dans la fonction `trac_sol_coup` à la place de celles désignées par les fichiers de données indiquées par les champs `donnees.nom_mail` et `donnees.nom_prep` qui sont alors facultatifs.

Les variables optionnelles $NFIG, VEC$ peuvent prendre d'autres valeurs avec un rôle analogue à celui joué dans la fonction `trac_solEc`. Pour l'utilisation de ces

variables et d'autres types d'appels on pourra se reporter au `help` de cette fonction `trac_sol_coup` et à la description des variables ci-dessous.

Variables d'entrée :

`donnees` : `STRUCT ARRAY`. Variable décrivant les données du problème EF. Pour les appels du type :

```
trac_sol_coup(donnees, Ac, Bc, tol, Uc, der, r, M) ;
trac_sol_coup(donnees, Ac, Bc, tol, Uc, der, r, M, nfig) ;
trac_sol_coup(donnees, Ac, Bc, tol, Uc, der, r, M, ...
               nfig, vec) ;
```

les champs à initialiser sont

```
donnees.D.E,
donnees.D.f,
donnees.nom_prep,
donnees.nom_mail.
```

Et pour les appels du type :

```
trac_sol_coup(donnees, Ac, Bc, tol, Uc, DER, R, M, nfig, ...
               vec, X, Y, T) ;
trac_sol_coup(donnees, Ac, Bc, tol, Uc, DER, R, M, nfig, ...
               vec, X, Y, T, Tc) ;
```

avec appel des variables de maillage `X`, `Y`, `T` et éventuellement de `Tc`, les champs à initialiser sont :

```
donnees.D.E,
donnees.D.f.
```

`Ac`, `Bc` : `DOUBLE ARRAY`. Vecteur lignes ou colonnes de longueurs identique. Coordonnées des points pour lesquels on calcule la valeur de la fonction ou de ses dérivées. On calcule aussi les valeurs de la fonction sur les sommets des subdivisions des segments $[(Ac(i), Bc(i)), (Ac(i+1), Bc(i+1))]$ en `M` segments. Les coupes sont tracées le long des segments formés par ces sommets.

`tol` : `DOUBLE ARRAY`. Scalaire qui indique une tolérance pour l'appartenance d'un point à un triangle. Il peut arriver qu'un point appartienne à un triangle, mais que la fonction `pt_tri` ne le signale pas en sortie à cause de l'erreur d'arrondi. On posera alors `tol` de l'ordre de l'erreur machine `eps` ou sinon à zéro.

`Uc` : `CELL ARRAY`. Vecteur colonne dont les cellules sont des vecteurs colonnes de type `DOUBLE ARRAY`. Représente les composantes dans sa base d'un élément de l'espace EF produit courant E (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des

composantes et valeur de la fonction aux *ddl* correspondants sont identiques. Les cellules *i* de `Uc` peuvent être vides si *i* n'est pas dans `r` ou `vec`. `Uc` peut aussi avoir des cellules vides.

`der` : DOUBLE ARRAY. Les valeurs de `der` = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable *x* et par rapport à la deuxième variable *y*.

`r` : DOUBLE ARRAY. Variable portant sur les composantes de l'espace EF produit *E*. Vecteur ligne, `r(i)` désigne la composante de la fonction à tracer. Si `r` = [] et si `vec` n'est pas utilisée ou est vide alors les cellules de `Uc` peuvent être toutes vides. La variable `r` doit être un vecteur ligne quand elle est utilisée avec `vec`.

`M` : DOUBLE ARRAY. Variable de raffinement. La coupe se fait suivant une subdivision des segments $[(Ac(i), Bc(i)), (Ac(i + 1), Bc(i + 1))]$ en *M* segments.

Variables d'entrée optionnelles :

Les noms des variables ci-dessous sont aussi les noms utilisés dans le programme :

Première variable optionnelle :

`nfig` : DOUBLE ARRAY ou STRUCT ARRAY. Variable de figure. Lorsque `nfig` est de type DOUBLE ARRAY, la composante `r(i)` de la fonction à tracer est tracée sur la figure numéro *i* si `nfig` = [] et sur la figure numéro `nfig-1+i` si `nfig` est un entier. Lorsque `nfig` est de type STRUCT ARRAY, `nfig.n` indique le numéro de figure sur lesquelles seront tracés tous les graphes dans un tableau de `nfig.t(1) × nfig.t(2)` figures.

Seconde variable optionnelle :

`vec` : DOUBLE ARRAY. Variable pour les tracés de champs de vecteur. Matrice de deux colonnes. Pour le tracé du champs de vecteurs donnés par les composantes `vec(k, 1)` et `vec(k, 2)` de la fonction représentée par `Uc`. Peut prendre la valeur [].

Troisième et quatrième variable optionnelle :

`X, Y` : DOUBLE ARRAY. Coordonnées des sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Cinquième variable optionnelle :

`T` : DOUBLE ARRAY. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

Sixième variable optionnelle :

`Tc` : CELL ARRAY. Vecteur ligne de cellules. Les cellules sont des matrices de type DOUBLE ARRAY. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du *i*-ème type d'espace d'EF choisi, cf. Tableau 2.2.3-1b du Paragraphe 2.2.3.

Variables de sortie :

U : **CELL ARRAY**. Vecteur colonne de même taille que **Uc**. **U{*i*}** est soit vide, soit, s'il est calculé, le vecteur colonne des valeurs aux points de la coupe de la *i*-ème composante de la fonction (ou d'une de ses dérivées, si **der** = 1 ou 2) représentée par **Uc{*i*}**.

A, B : **DOUBLE ARRAY**. Vecteurs lignes représentant les coordonnées des points de la coupe.

Exemple 1 : Cet exemple est tiré du **help** de la fonction **trac_sol_coup**. On considère le maillage du carré unité en deux triangles de la figure 4.3-7 et l'espace EF produit (restreint ici à un seul espace) $E = P_{1,3}^c = P_1 \oplus \text{bulle } P_3$. Cet exemple trace le graphe des deux fonctions bulles, effectue une coupe suivant la droite $y = 1/3$, et demande à l'utilisateur d'indiquer sur le graphique des fonctions bulles, deux points pour faire une coupe suivant la droite passant par ces deux points. On notera l'utilisation de la fonction **plot** utilisée pour un tracé de coupe en 2D.

EXEMPLE sur deux triangles :

```
close all
clear
T = [1,2,3;2,3,4] ;
X = [0,1,0,1] ;
Y = [0,0,1,1] ;
Ac = 0 :0.1 :1 ;
Bc = 1/3*ones(1,11) ;
Z = pt_tri(Ac, Bc, T, X, Y) ;
donnees.D.E = 13 ;
donnees.D.f = 1 ;
der = 0 ;
r = 1 ;
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees, X, Y, T, []) ;
Uc{1}(5,1) = 1 ;
Uc{1}(6) = 1 ;
NFIG = 1 ;
tol = 0 ;
[V,A,B] = trac_sol_coup(donnees, Ac, Bc, tol, Uc, ...
                        der, r, 20, NFIG, [], X, Y, T, Tc) ;
figure(2)
plot(A,V{1})
```

```

NFIG = 3 ;
[V2,A2,B2] = trac_solEc(donnees, Uc, der, r, 40, ...
                        NFIG, [], X, Y, T, Tc) ;

view(2)
[A3,B3] = GINPUT(2) ;
close(NFIG)
[V,A4,B4] = trac_sol_coup(donnees, A3, B3, tol, Uc, ...
                          der, r, 20, NFIG, [], X, Y, T, Tc) ;

```

Algorithme :

EF/trace_graphe_interp/trac_sol_coup.m

On remet ici la partie déclarative de la fonction :

```

function [U, A, B] = trac_sol_coup(donnees, ...
                                   Ac, Bc, tol, Uc, der, r, M, varargin) ;

```

L'algorithme de cette fonction est basée sur l'utilisation de la fonction `pt_tri` pour connaître à quels triangles appartiennent les points de la coupe et de la fonction `valeur_fonc_EF` pour connaître la valeur des fonctions en ces points.

Fonctions de Assemblage_elementaire.

Ci-dessous, les fonctions de ce répertoire se répartissent de la manière suivante :

Fonctions sur les coefficients des formules de quadrature

assemble_coeff_T (triangles)

assemble_coeff_A (arêtes frontière)

Valeurs des fonctions de bases, pour le fichier relatif aux espaces d'EF.

matrice_elem_prepa_T (triangles)

matrice_elem_prepa_A (arêtes frontière)

matrice_elem_prepa_Ad (arêtes intérieures et frontière)

Calcul des matrices élémentaires des systèmes.

matrice_elem_T (triangles)

matrice_elem_A (arêtes frontière)

matrice_elem_Ad (arêtes du maillage, en développement, fonctionne)

Calcul des matrices élémentaires des systèmes.

matrice_elem_T_scm (triangles)

matrice_elem_A_scm (arêtes frontière)

matrice_elem_Ad_scm (arêtes du maillage, en développement, fonctionne)

matrice_elem_Ad_int_LR (méthode de Lesaint-Raviart)

Calculs intermédiaires avant l'assemblage.

pre_assemble_T. Utilisé par matrice_elem_prepa_T et matrice_elem_T.

pre_assemble_A. Utilisé par :

matrice_elem_prepa_A,

matrice_elem_A,

matrice_elem_prepa_Ad et

matrice_elem_Ad_int_LR.

Fonctions communes aux précédente utilisées pour le calcul système ou second membre.

equationJ

assemble_equ

Valeur des fonctions de base.

`F_base_T` (triangles)

`F_base_A` (arêtes)

Actuellement ces deux fonctions sont identiques. Elles font appel à la fonction

`F_base`

Autres fonctions : `valeur_fonc_quad_K`, `F_base_der2`, `F_base_der2_3` : non utilisées dans le code.

Ci-dessous, les éléments finis considérés seront le triplet dont le premier terme est un triangle K , le second terme un espace de la forme (*cf.* Paragraphe 1.4, Eqn. (1.20))

$$P_{i,j}(K) = \mathcal{A}_i(K) \oplus \mathcal{B}_j(K)$$

et le troisième terme, l'ensemble de formes linéaires

$$\Sigma_d = \{\sigma_m, 1 \leq m \leq \dim(P_{i,j}(K)), \sigma_m(f) = f|_K(x_m), \forall f \in P_{i,j}(K)\},$$

pour le cas discontinu et

$$\Sigma_c = \{\sigma_m, 1 \leq m \leq \dim(P_{i,j}(K)), \sigma_m(f) = f(x_m), \forall f \in P_{i,j}(K)\},$$

pour le cas continu, l'ensemble des points $\{x_m\}_{1 \leq m \leq \dim(P_{i,j}(K))}$ étant les *ddl* de l'espace $P_{i,j}(K)$ défini au Paragraphe 2.3.3.

Les éléments finis considérés seront alors soit l'EF $P_{i,j}^d(K)$ (discontinu) soit l'EF $P_{i,j}^c(K)$ (continu).

Les espace d'EF E_i dans le code seront alors de la forme (*cf.* Paragraphe 1.4, Eqns. (1.21), (1.22)) :

$$\begin{aligned} P_{i,j}^d &= \{v \in L^2(\Omega), v|_K \in P_{i,j}(K), \forall K \in \mathcal{T}, (i,j) \in \mathbb{N}^2, \\ P_{i,j}^c &= \{v \in \mathcal{C}^0(\bar{\Omega}), v|_K \in P_{i,j}(K), \forall K \in \mathcal{T}, (i,j) \in \mathbb{N}^* \times \mathbb{N}, \end{aligned}$$

avec dans le code $i \leq 10$ et $j \leq 10$.

Ci-dessous, pour un triangle K , E_K désignera l'espace d'EF $P_{i,j}^d$ ou $P_{i,j}^c$ obtenu pour $\Omega = K$.

4.3. DESCRIPTION DES FONCTIONS F^N Assemblage_elementaire/pre_assemble_T

Fonction :

```
function M = pre_assemble_T(omega, espace, der, a1, b1, c1, ...
    a2, b2, c2, de, A, B, coeff, cas, D, L1, L2, L3) ;
```

Calculs préparatoires pour le calcul de matrices élémentaires d'assemblage sur un triangle.

Remarque. Dans la version actuelle de `pre_assemble_T` et de la fonction `F_base_T` appelée par cette fonction, `pre_assemble_T` ne dépend pas des variables `A`, `B` et `D` qui peuvent être mises à [].

Nous décrivons la fonction `pre_assemble_T` suivant les quatre valeurs prises par `cas` (exemples 1a, 1b, 2 et 3 tirés du `help` de `pre_assemble_T`) puis nous donnons le lien entre cette fonction et la formule (3.13) du Paragraphe 3.1.2.1 pour le calcul des matrices élémentaires (exemples 4a et 4b tirés du `help` de `pre_assemble_T`).

Les exemples qui suivent sont tirés du `help` de la fonction `pre_assemble_T`.

- Pour `cas = 1`. Duplication de vecteur `coeff`.

```
M = pre_assemble_T([], espace, [], [], [], [], [], ...
    [], [], [], [], [], coeff, 1, [], [], [], []);
```

calcule, pour un espace d'EF sur un triangle K noté E_K et de numéro `espace` la matrice `M` de $\dim(E_K)$ lignes où chaque ligne de `M` est égale au vecteur ligne `coeff`.

EXEMPLE 1a. Pour le $P_3(K)$ continu de dimension 10.

```
clear
cas = 1;
espace = 3;%P3 continu
coeff = 1 :11;
M = pre_assemble_T([], espace,[], [], [], [], [], [], ...
    [], [], [], [], coeff, cas, [], [], [], []);
```

Cet exemple donne

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \end{pmatrix}.$$

En pratique, `coeff` est le vecteur des coefficients de la formule d'intégration numéro `D.intT`, comme dans l'exemple qui suit, où `D` est la variable qui décrit

4. DESCRIPTION DU CODE FONCTION `Assemblage_elementaire/pre_assemble_T`

l'espace EF produit E utilisé ainsi que les formules de quadrature. Dans le code la variable `D` est donnée par la valeur de `donnees.D` où `donnees` est la variable de type `struct` qui décrit la structure de données du problème EF (*cf.* le Paragraphe 2.2) et où le champ `D` est décrit au Tableau 2.2.1-1-c, Paragraphe 2.2.1.

EXEMPLE 1b. Pour le $P_3(K)$ continu, de dimension 10 et avec `coeff` tiré d'une formule d'intégration. Ici `coeff` est indépendant des coordonnées du triangle données par `X` et `Y`.

```
clear
cas = 1 ;
espace = 3 ;
X = [0,1,0] ;
Y = [0,0,1] ;
D.intT = 4 ;
[A1, B1, C1, A2, B2, C2, DE, A, B, coeff, L1, L2, L3] ...
    = assemble_coeff_T(X, Y, D) ;
M = pre_assemble_T([], espace, [], [], [], [], [], [], ...
    [], [], [], [], coeff, cas, [], [], [], []) ;
```

Cet exemple donne avec

$$\text{coeff} = \frac{1}{90} (0 \ 4 \ -1 \ 4 \ 0 \ 4 \ 8 \ 8 \ 4 \ -1 \ 8 \ -1 \ 4 \ 4 \ 0)$$

qui représente les coefficients de la formule de quadrature numéro 4 du (3.24) du Paragraphe 3.3.1, la valeur

$$M = \frac{1}{90} \begin{pmatrix} 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \\ 0 & 4 & -1 & 4 & 0 & 4 & 8 & 8 & 4 & -1 & 8 & -1 & 4 & 4 & 0 \end{pmatrix}.$$

- Pour `cas = 2`. Dérivation de la fonction représentée par `omega`.

```
M = pre_assemble_T(omega, espace, der, a1, b1, c1, a2, ...
    b2, c2, de, A, B, coeff, 2, D, L1, L2, L3) ;
```

effectue le calcul suivant. Soit E_K l'espace d'EF sur le triangle K de sommets $(a1, a2)$, $(b1, b2)$, $(c1, c2)$ tel que E_K a pour numéro `espace`. Soit ω la fonction

4.3. DESCRIPTION DES FONCTIONS F^N `Assemblage_elementaire/pre_assemble_T`

dont les composantes dans la base de E_K sont données par le vecteur ligne - ou colonne - `omega` de taille $\dim(E_K)$. La variable `M` est alors un vecteur ligne de taille `length(L1)` et cette variable représente les valeurs de la dérivée partielle `der` = 0, 1 ou 2 de ω aux points de coordonnées barycentriques relatives au triangle de sommets (a_1, a_2) , (b_1, b_2) et (c_1, c_2) données par les vecteurs lignes de tailles identiques `L1`, `L2` et `L3`.

La variable `de` est le Jacobien de la bijection affine du triangle de référence $(0, 0)$, $(1, 0)$, $(0, 1)$ sur le triangle K :

$$\text{de} = [a_1, b_1, c_1] * [b_2 - c_2, -a_2 + c_2, -b_2 + a_2]',$$

En général les points donnés par `L1`, `L2` et `L3` sont issus d'une des formules de quadrature utilisée pour l'intégration sur le triangle de sommets de coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) et les vecteurs lignes `A`, `B` sont alors les coordonnées en x et y de ces points. Le numéro de la formule de quadrature utilisée est donné par la valeur du champ `D.intT`. Les vecteurs `A` et `B` et la variable `D` sont actuellement facultatifs et peuvent être mis à `[]` : la fonction ne dépend pas de ces variables. Pour la valeur de `cas` = 2, 3, ou 4 on peut aussi poser `coeff` = `[]`, le résultat étant indépendant de cette variable.

Pour un triangle de coordonnées `X` et `Y` les valeurs ci-dessus peuvent être données par la commande

```
[a1, b1, c1, a2, b2, c2, de, A, B, coeff] = ...
assemble_coeff_T(X, Y, D) ;
```

avec

$$a_1 = X(1), \quad b_1 = X(2), \quad c_1 = X(3), \quad a_2 = Y(1), \quad b_2 = Y(2), \quad c_2 = Y(3).$$

Remarque. Pour `der` = 0 dans l'appel ci-dessus de la fonction `pre_assemble_T`, le résultat est indépendant de `A1`, `B1`, `C1`, `A2`, `B2`, `C2`, `de` car alors il ne dépend que des valeurs `L1`, `L2` et `L3` qui sont constantes pour chacune des formules de quadrature `assemble_coeff_T` programmées.

EXEMPLE 2. Pour le $P_1(K) \oplus \text{bulle}P_3(K)$ – continu, de dimension 4 et avec `coeff` tiré d'une formule d'intégration. Ici `coeff` est indépendant des coordonnées du triangle données par `X`, `Y` (comme indiqué ci-dessous le résultat ne dépend pas de `coeff` que l'on peut poser à `[]`).

```
clear
omega = [1,2,3,0] ; %Fonction 1 + x + 2*y
cas = 2 ;
espace = 13 ;
der = 0 ;
X = [0,1,0] ;
```

4. DESCRIPTION DU CODE FONCTION Assemblage_elementaire/pre_assemble_T

```

Y = [0,0,1] ;
D.intT = 4 ;
[A1, B1, C1, A2, B2, C2, de, A, B, coeff, L1, L2, L3] ...
                                = assemble_coeff_T(X, Y, D) ;

%Valeur de omega : DER = 0 :
M1 = pre_assemble_T(omega, espace, der, A1, B1, C1, A2, B2, ...
                    C2, de, [], [], coeff, cas, [], L1, L2, L3) ;

%Valeur de omega,x : DER = 1 :
der = 1 ;
M2 = pre_assemble_T(omega, espace, der, A1, B1, C1, A2, B2, ...
                    C2, de, [], [], coeff, cas, [], L1, L2, L3) ;

```

Ci-dessus ω représente les composantes dans la base canonique choisie pour E_K de la fonction $u : (x, y) \rightarrow 1 + x + 2y$ qui ici est dans E_K .

On obtient pour les coordonnées cartésiennes des 15 points de la formule de quadrature :

$$A = \frac{1}{4} (0 \ 1 \ 2 \ 3 \ 4 \ 0 \ 1 \ 2 \ 3 \ 0 \ 1 \ 2 \ 0 \ 1 \ 0) ;$$

$$B = \frac{1}{4} (0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 3 \ 3 \ 4) ,$$

On trouve pour M1 les valeurs de la fonction représentée par ω donc ici de u aux points de la formule de quadrature :

$$M1 = 1 + A + 2B = \frac{1}{4} (4 \ 5 \ 6 \ 7 \ 8 \ 6 \ 7 \ 8 \ 9 \ 8 \ 9 \ 10 \ 10 \ 11 \ 12)$$

et pour M2 les valeurs les valeurs aux points de la formule de quadrature de la dérivée partielle par rapport à x de fonction représentée par ω qui ici est $u_{,1} = 1$:

$$M2 = (1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1) .$$

- Pour $\text{cas} = 3$. Valeur des fonctions de base.

```

M = pre_assemble_T([], espace, der, a1, b1, c1, a2, b2, c2, ...
                    de, A, B, coeff, 3, D, L1, L2, L3) ;

```

calcule les valeurs de la dérivée $\text{der} = 0, 1$ ou 2 des fonctions de base de l'espace E_K de numéro espace . Les valeurs sont données aux points de coordonnées barycentriques $L1, L2$ et $L3$ définies ci-dessus pour $\text{cas} = 2$ avec les mêmes remarques ci-dessus pour A, B et D qui sont facultatives. La variable M est la matrice $\text{length}(L1) \times \dim(E_K)$ des valeurs des fonctions de base telle que $M(i, j)$ est la valeur de la dérivée partielle $\text{der} = 0, 1$ ou 2 de la j -ème fonction de base de E_K au point de coordonnées barycentriques $L1(i), L2(i)$ et $L3(i)$.

4.3. DESCRIPTION DES FONCTIONS F^N `Assemblage_elementaire/pre_assemble_T`

EXEMPLE 3. Pour le $P_1(K)$ continu, de dimension 3 et avec `coeff` tiré d'une formule d'intégration. Ici `coeff` est indépendant des coordonnées du triangle données par X, Y .

```
clear
cas = 3 ;
espace = 1 ;
der = 0 ;
X = [0,1,0] ;
Y = [0,0,1] ;
D.intT = 4 ;
[A1, B1, C1, A2, B2, C2, de, A, B, coeff, L1, L2, L3] ...
    = assemble_coeff_T(X, Y, D) ;
M = pre_assemble_T([], espace, der, A1, B1, C1, A2, B2, C2, ...
    de, A, B, coeff, cas, D, L1, L2, L3) ;
```

On obtient pour M les valeurs des fonctions de base qui sont ici $1 - x - y, x, y$ aux points de la formule de quadrature.

$$M = [1 - A' - B', A', B] = \begin{pmatrix} 1 & 0 & 0 \\ 0.75 & 0.25 & 0 \\ 0.5 & 0.5 & 0 \\ 0.25 & 0.75 & 0 \\ 0 & 1 & 0 \\ 0.75 & 0 & 0.25 \\ 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0 & 0.75 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \\ 0 & 0.5 & 0.5 \\ 0.25 & 0 & 0.75 \\ 0 & 0.25 & 0.75 \\ 0 & 0 & 1 \end{pmatrix}.$$

- Pour `cas = 4`. Valeur des fonctions de base. Ce cas est identique au cas précédent. Excepté la matrice de sortie M qui est transposée et donc de taille $\dim(E_K) \times \text{length}(L1)$.

Utilisation de la fonction `pre_assemble_T` pour l'assemblage de matrices élémentaires. Dans ce Paragraphe on met en œuvre avec `pre_assemble_T` la formule (3.13) du Paragraphe 3.1.2.1 pour le calcul des matrices élémentaires. Dans le code le calcul des matrices élémentaire est effectué avec la fonction `pre_assemble_T` sur le mode de calcul présenté dans les deux exemples qui suivent.

4. DESCRIPTION DU CODE FONCTION `Assemblage_elementaire/pre_assemble_T`

Soit X et Y les coordonnées d'un triangle K . Soit `espaceom`, `espace1`, `espace2` les numéros de trois espaces d'EF sur K notés E_ω , E_1 et E_2 . Soit `omega` un vecteur ligne (ou colonne) représentant les composantes dans la base de E_ω d'une fonction $\omega \in E_\omega$. Soit $\{\Phi_j\}_{1 \leq j \leq \dim(E_1)}$ la base de E_1 , $\{\Psi_j\}_{1 \leq j \leq \dim(E_2)}$ celle de E_2 . Alors la matrice L d'assemblage élémentaire de dimension

$$\dim(E_2) \times \dim(E_1)$$

et de terme (p, q) défini par :

$$L(p, q) = \int_K \omega_{,der}(x, y) \Phi_{q,der_1}(x, y) \Psi_{p,der_2}(x, y)$$

est donnée, à l'erreur d'intégration près, par le produit (*cf.* (3.13) du Paragraphe 3.1.2.1) :

$$L = (C * \text{diag}(OM) .* V) * U;$$

ou encore de manière équivalente par :

$$L = (C .* \text{repmat}(OM, [\text{size}(C, 1), 1])). * V) * U;$$

les variables C , OM , U , V étant données par :

- Informations relatives au triangle et à la formule de quadrature :

$$[a1, b1, c1, a2, b2, c2, de, A, B, coeff, L1, L2, L3] = \dots$$

$$\text{assemble\_coeff\_T}(X, Y, D);$$

`cas = 1` donne C :

$$C = \text{pre\_assemble\_T}([], \text{espace2}, [], [], [], [], [], [], \dots$$

$$[], [], [], [], \text{coeff}, 1, D, [], [], []);$$

`cas = 2` donne OM :

$$OM = \text{pre\_assemble\_T}(\text{omega}, \text{espaceom}, \text{derom}, a1, b1, c1, a2, \dots$$

$$b2, c2, de, A, B, \text{coeff}, 2, D, L1, L2, L3);$$

`cas = 3` donne U :

$$U = \text{pre\_assemble\_T}([], \text{espace1}, \text{der1}, a1, b1, c1, a2, b2, \dots$$

$$c2, de, A, B, \text{coeff}, 3, D, L1, L2, L3);$$

`cas = 4` donne V :

$$V = \text{pre\_assemble\_T}([], \text{espace2}, \text{der2}, a1, b1, c1, a2, b2, \dots$$

$$c2, de, A, B, \text{coeff}, 4, D, L1, L2, L3);$$

où $(a1, a2)$, $(b1, b2)$, $(c1, c2)$ sont les coordonnées des sommets du triangle K , D la variable donnant le numéro des formules d'intégration.

4.3. DESCRIPTION DES FONCTIONS `FN Assemblage_elementaire/pre_assemble_T`

On remarque que dans le calcul de `C` et de `V` les numéros d'espace doivent être identiques pour que `C` et `V` soient de même taille. Cette taille entre en compte notamment dans la compatibilité des termes du produit :

$$L = (C * \text{diag}(OM) .* V) * U;$$

avec `OM` vecteur ligne.

Dans le code, les opérations de calculs des matrices élémentaires comme la matrice `L` ci-dessus sont effectués par l'intermédiaire de la fonction `matrice_elem_T`. Comme indiqué au Paragraphe 3.1.2.1, le calcul et le stockage des termes du type `U` ou `V` ci-dessus est effectués par la fonction `EF_prepa_don` par l'intermédiaire de la fonction `matrice_elem_prepa_T`.

EXEMPLE 4a. Pour l'assemblage de la forme bilinéaire :

$$(u, v) \rightarrow \int_K (1 + x + 2y)u(x, y)v(x, y)dxdy$$

définie sur l'espace `EF` produit :

$$E = E_1 \times E_2 = P_2(K) - \text{continu} \times P_3(K) - \text{continu},$$

avec `K` triangle de sommets $(0, 0)$, $(1, 0)$, $(0, 1)$. On choisit comme espace `Eω` : `Eω = P1(K)` et $\omega(x, y) = 1 + x + 2y$. `L` se calcule alors par :

```
clear
X = [0,1,0] ;
Y = [0,0,1] ;
D.intT = 4 ;
%Composantes sur le P1(K) de omega :
omega = [1,2,3] ;
espaceom = 1 ;
derom = 0 ;
espace1 = 2 ;
der1 = 0 ;
espace2 = 3 ;
der2 = 0 ;
[A1, B1, C1, A2, B2, C2, de, A, B, coeff, L1, L2, L3] = ...
    assemble_coeff_T(X, Y, D) ;
C = pre_assemble_T([], espace2, [], [], [], [], [], [], ...
    [], [], [], [], coeff, 1, D, L1, L2, L3) ;
```

4. DESCRIPTION DU CODE FONCTION Assemblage_elementaire/pre_assemble_T

```
%Valeur aux point d'integration de omega (derivee d'ordre 0) :
OM = pre_assemble_T(omega, espaceom, derom, A1, B1, C1, ...
                    A2, B2, C2, de, A, B, coeff, 2, D, L1, L2, L3);

%Calcul de U : Valeur des fonctions de base du P2 aux points
%de la formule de quadrature :
U = pre_assemble_T([], espace1, der1, A1, B1, C1, A2, ...
                    B2, C2, de, A, B, coeff, 3, D, L1, L2, L3);

%Calcul de V : valeur des fonctions de base du P3 aux points de
%la formule de quadrature :
V = pre_assemble_T([], espace2, der2, A1, B1, C1, A2, B2, ...
                    C2, de, A, B, coeff, 4, D, L1, L2, L3);

%Matrice L :
L = (C*diag(OM).*V)*U;

%Ou bien par duplication de OM :
OM2 = repmat(OM, [size(C, 1), 1]);
LD = (C.*OM2.*V)*U;
```

EXEMPLE 4b. On reprend l'exemple précédent en modifiant E_ω par $E_\omega = P_2(K)$ et en conservant $\omega(x, y) = 1 + x + 2y$. On doit alors retrouver le même résultat. Ensuite, pour illustrer le cas où le poids n'est pas forcément dans un espace d'EF, on effectue aussi le calcul avec $E_\omega = P_0(K)$ et le choix $\omega(x, y) = 2$ qui est alors l'interpolé de $1 + x + 2y$ au centre de gravité. On obtient alors une approximation de L.

```
clear
X = [0,1,0];
Y = [0,0,1];
D.intT = 4;
derom = 0;
espace1 = 2;
der1 = 0;
espace2 = 3;
```

4.3. DESCRIPTION DES FONCTIONS F^N Assemblage_elementaire/pre_assemble_T

```

der2 = 0 ;
[A1, B1, C1, A2, B2, C2, de, A, B, coeff, L1, L2, L3] = ...
                                assemble_coeff_T(X, Y, D) ;
C = pre_assemble_T([], espace2, [], [], [], [], [], [], ...
                    [], [], [], [], coeff, 1, D, L1, L2, L3) ;
U = pre_assemble_T([], espace1, der1, A1, B1, C1, A2, ...
                    B2, C2, de, A, B, coeff, 3, D, L1, L2, L3) ;
V = pre_assemble_T([], espace2, der2, A1, B1, C1, A2, B2, ...
                    C2, de, A, B, coeff, 4, D, L1, L2, L3) ;
%Composantes sur le P2(K) de omega
omega = [1,2,3,1.5,2,2.5] ;
espaceom = 2 ;
OM = pre_assemble_T(omega, espaceom, derom, A1, B1, C1, ...
                    A2, B2, C2, de, A, B, coeff, 2, D, L1, L2, L3) ;
%Matrice L :
LP2 = (C*diag(OM).*V)*U ;
%Composantes de l'interpole au centre de gravite sur le P0(K)
% de omega :
omega = [2] ;
espaceom = 100 ;
OM = pre_assemble_T(omega, espaceom, derom, A1, B1, C1, ...
                    A2, B2, C2, de, A, B, coeff, 2, D, L1, L2, L3) ;
%Matrice L :
LP0 = (C*diag(OM).*V)*U ;

```

Variables d'entrée :

omega : DOUBLE ARRAY.

Pour **cas** = 2 : Vecteur ligne ou colonne de taille $\dim(E_K)$ où E_K est l'espace d'EF sur un triangle K , de numéro d'espace d'EF **espace**. Ce vecteur représente alors les composantes d'une fonction dans la base de l'espace d'EF désigné par la variable **espace** décrite ci-après.

Pour **cas** = 1, 2 ou 4 : le résultat donné par la fonction est indépendant de **omega** qui peut être mis à [].

espace : DOUBLE ARRAY. Numéro d'espace d'EF sur un triangle K . Ces espaces sont décrits au Paragraphe 1.4. Les numéros d'espaces sont actuellement :

4. DESCRIPTION DU CODE FONCTION `Assemblage_elementaire/pre_assemble_T`

1 à 9 : éléments P_1 à P_9 continus, 200 : P_{10} continu, 10 à 99 : éléments continus à «bulles», exemples :

13 : $P_1 \oplus$ bulle P_3 ,

14 : $P_1 \oplus 3$ bulles P_4 ,

23 : $P_1 \oplus$ bulle P_3 ,

24 : $P_3 \oplus 3$ bulles P_4 somme non directe.

Ensuite, 100 à 109 : éléments P_0 à P_9 discontinus, 300 : P_{10} discontinu, 110 à 199 : éléments discontinus à «bulles», exemple :

113 : $P_1 \oplus$ bulle P_3 ,

114 : $P_1 \oplus 3$ bulles P_4 ,

123 : $P_1 \oplus$ bulle P_3 ,

124 : $P_3 \oplus 3$ bulles P_4 somme non directe.

der : `DOUBLE ARRAY`. Les valeurs de **der** = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

a1, b1, c1, a2, b2, c2 : `DOUBLE ARRAY`. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K .

de : `DOUBLE ARRAY`. scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

A, B : `DOUBLE ARRAY`. Vecteurs lignes désignant, en général, les coordonnées en x et y des points de coordonnées barycentriques **L1**, **L2** et **L3**. La fonction ne dépend pas actuellement de ces variables qui peuvent être mises à [].

coeff : `DOUBLE ARRAY`. Vecteur ligne des coefficients de la formule de quadrature. Ici, il peut aussi s'agir d'un vecteur quelconque à dupliquer, la fonction est indépendante de **coeff** qui peut valoir [] lorsque **cas** = 2, 3 ou 4.

cas : `DOUBLE ARRAY`. scalaire prenant les valeurs 1, 2, 3, 4. Choix du type de terme à calculer : duplication de **coeff** (**cas** = 1), dérivation de la fonction ω représentée par **omega** (**cas** = 2) et calcul des fonctions de base et de leurs dérivées (**cas** = 3 ou 4). La valeur **cas** = 4 donne la matrice transposée du cas où **cas** = 3.

D : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à **donnees.D**. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe

4.3. DESCRIPTION DES FONCTIONS F^N `Assemblage_elementaire/pre_assemble_T`

2.2.1. Cette variable n'intervient pas actuellement dans cette fonction, elle peut être mise à [].

Remarque. Dans les exemples ci-dessus, cette variable est utilisée dans la fonction `assemble_coeff_T` par l'intermédiaire de son champ `D.intT` qui donne le numéro de la formule de quadrature.

`L1, L2, L3 : DOUBLE ARRAY`. Vecteurs lignes de tailles identiques de coordonnées barycentriques relativement au triangle de sommets de coordonnées $(a1, a2)$, $(b1, b2)$, $(c1, c2)$. En général les points donnés par `L1, L2` et `L3` sont issus de l'une des formules de quadrature utilisée pour l'intégration sur le triangle et `L1, L2, L3` sont alors de longueur égale au nombre de points de la formule de quadrature.

Variables de sortie :

`M : DOUBLE ARRAY`. Sa valeur dépend de `cas`. Notons E_K l'espace d'EF sur un triangle K , de numéro d'espace d'EF `espace`.

Pour `cas = 1`, `M` est une matrice de taille $\dim(E_K) \times \text{length}(\text{coeff})$. Chaque ligne de `M` contient la valeur `coeff`.

Pour `cas = 2`, `M` est un vecteur ligne de longueur `length(L1)`. Soit ω la fonction ayant pour composantes le vecteur (ligne -ou colonne) `omega` dans la base de E_K . Alors `M` représente les valeurs de ω_{der} , `der = 0, 1, 2`, aux points de coordonnées barycentriques `L1, L2` et `L3` vecteurs lignes provenant en général d'une formule de quadrature.

Pour `cas = 3`, `M` est la matrice de $\text{length}(L1) \times \dim(E_K)$ des valeurs des fonctions de base telle que `M(i, j)` est la valeur de la dérivée partielle `der = 0, 1` ou `2` de la j -ème fonction de base de E_K au point de coordonnées barycentriques `L1(i), L2(i)` et `L3(i)`.

Pour `cas = 4`, `M` est la matrice transposée du cas précédent.

Variables facultatives : Dans sa version actuelle la fonction ne dépend pas de ces variables qui peuvent être mises à []. Ce sont `D, A` et `B`.

Exemple : On pourra consulter les exemples ci-dessus qui sont aussi donnés dans le `help` de la fonction.

Algorithme :

`EF/Assemblage_elementaire/pre_assemble_T.m`

On remet ici la partie déclarative de la fonction :

```
function M = pre_assemble_T(omega, espace, der, a1, b1, c1, ...  
                           a2, b2, c2, de, A, B, coeff, cas, D, L1, L2, L3) ;
```

L'algorithme est basé sur l'utilisation de la fonction `F_base_T` décrite ci-après. La fonction `F_base_T` utilise elle-même la fonction `F_base` qui est la fonction principale

4. DESCRIPTION DU CODE FONCTION `Assemblage_elementaire/pre_assemble_A`

pour le calcul des valeurs des fonctions de base en des points du plan dont la position est donnée par leur coordonnées barycentriques.

Suivant la valeur de `cas` :

- Duplication des points de la formule de quadrature :

Cas 1

```
[Z, dim] = F_base_T(espace, [], [], [], [], [], [], [], ...  
                    [], [], [], D, [], [], []);  
  
M = coeff(ones(1,dim), :);
```

- Calcul des composantes de la fonction représentée par `omega` :

Cas 2

```
[Z, dim] = F_base_T(espace, der, A, B, a1, a2, b1, b2, ...  
                    c1, c2, de, D, L1, L2, L3);  
  
M = sum(diag(omega)*Z, 1);
```

- Calcul de matrice d'assemblage :

Cas 3

```
[M, dim] = F_base_T(espace, der, A, B, a1, a2, b1, b2, ...  
                    c1, c2, de, D, L1, L2, L3);  
  
M = M';
```

- Calcul de matrice d'assemblage, la matrice calculée est la transposée de la matrice du cas précédent

Cas 4

```
[M, dim] = F_base_T(espace, der, A, B, a1, a2, b1, b2, ...  
                    c1, c2, de, D, L1, L2, L3);
```

Fonction :

```
function M = pre_assemble_A(omega, espace, der, a1, b1, c1, ...  
                            a2, b2, c2, de, A, B, coeff, cas, D, L1, L2, L3);
```

Calculs préparatoires pour le calcul de matrices élémentaires d'assemblage sur les arêtes. Dans sa version actuelle cette fonction est identique à `pre_assemble_T`, les appels de la fonction `F_base_T` étant remplacés par des appels de la fonction `F_base_A` identique elle-même à `F_base_T` dans la version actuelle du code.

Remarque. Comme pour les fonctions `pre_assemble_T` et `F_base_T`, les fonctions `pre_assemble_A` et `F_base_A`, dans leur version actuelle, ne dépendent pas des variables `A`, `B` et `D` qui peuvent être posées à `[]`.

Fonction :

```
function [z, dim] = F_base_T(e, d, A, B, a1, a2, b1, b2, c1, ...
                             c2, de, D, L1, L2, L3) ;
```

Valeur des fonctions de base d'un espace d'EF sur un triangle K . Les valeurs sont données en des points repérés par leur coordonnées barycentriques. Ces points sont par exemple ceux issus d'une formule de quadrature sur les triangles.

Remarque. Dans sa version actuelle, `F_base_T` ne dépend pas des variables `A`, `B` et `D` qui peuvent être mises à []. L'appel

```
[Z, dim] = F_base_T(espace, der, Ac, Bc, a1, a2, b1, b2, ...
                    c1, c2, de, D, L1, L2, L3) ;
```

(avec `der` pour `d` dans la déclaration de la fonction) renvoie les valeurs des fonctions de base et de leurs dérivées de l'espace d'EF E_K (sur un triangle K) de numéro `espace`. Les valeurs sont données aux points de coordonnées barycentriques relatives au triangle K de sommets $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$ et qui sont données par les vecteurs lignes `L1`, `L2`, `L3`. La dérivation est indiquée par `der` = 0, 1 ou 2 et `de` est le Jacobien donné par

$$de = [a1, b1, c1] * [b2 - c2, -a2 + c2, -b2 + a2]'.$$

En général les points donné par `L1`, `L2` et `L3` sont issus d'une des formules de quadrature utilisée pour l'intégration sur le triangle de sommets de coordonnées $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$ et ces points ont pour coordonnées en x et y les vecteurs `Ac`, `Bc`. Le numéro de la formule de quadrature utilisée est donné par la valeur du champ `D.intT`. Dans sa version actuelle, `F_base_T` ne dépend pas de ces vecteurs `Ac` et `Bc` et de `D` qui sont facultatifs et peuvent être mis à [].

En sortie, `dim` est la dimension de E_K l'espace d'EF de numéro `espace` et `Z` est la matrice $\dim(E_K) \times \text{length}(L1)$ des valeurs des fonctions de base telle que $Z(i, j)$ est la valeur de la dérivée partielle `der` = 0, 1 ou 2 de la i -ème fonction de base de E_K au point de coordonnées barycentriques $L1(j)$, $L2(j)$ et $L3(j)$.

La commande

```
[Z, dim] = F_base_T(espace, [], [], [], [], [], [], [], [], [], [], [], [], []);
```

renvoie `Z` = [] et la valeur de `dim`, la dimension de l'espace d'EF de numéro donné par `espace`. C'est aussi le cas dès qu'un des variables `L1`, `L2` ou `L3` prend la valeur []. Si `espace` n'est pas un numéro d'espace d'EF programmé dans le code, `dim` n'est pas initialisée.

Variables d'entrée :

espace : DOUBLE ARRAY. Numéro d'espace d'EF sur un triangle K . Ces espaces sont décrits au Paragraphe 1.4. La description de cette variable est donnée à la fonction `F_base_T` ou `pre_assemble_T`, cf. aussi les exemples donnés dans ces deux fonctions.

d noté aussi **der** dans le commentaire ci-dessus :

d : DOUBLE ARRAY. Les valeurs de $d = 0, 1, 2$ indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature. La fonction ne dépend pas actuellement de ces variables qui peuvent être mises à `[]`.

a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K .

de : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

D : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe 2.2.1. La fonction ne dépend pas actuellement de cette variable qui peut être mise à `[]`.

L1, L2, L3 : DOUBLE ARRAY. Vecteurs lignes de tailles identiques de coordonnées barycentriques relativement au triangle de sommets de coordonnées $(a1, a2)$, $(b1, b2)$, $(c1, c2)$.

Variables de sortie :

Z : DOUBLE ARRAY. Matrice de $\dim(E_K) \times \text{length}(L1)$ des valeurs des fonctions de base d'un espace d'EF E_K sur K , telle que $Z(i, j)$ est la valeur de la dérivée partielle $der = 0, 1$ ou 2 de la i -ème fonction de base de E_K au point de coordonnées barycentriques $L1(j)$, $L2(j)$ et $L3(j)$. **Z** vaut `[]` si une des variables **L1**, **L2** ou **L3** prend la valeur `[]`.

Variables facultatives : Dans sa version actuelle la fonction ne dépend pas de ces variables qui peuvent être mises à `[]`. Ce sont **D**, **Ac** et **Bc**.

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base_T

Les exemples suivants sont aussi donnés dans le `help` de la fonction `F_base_T`.

Exemple 1 : Cet exemple traite le cas des valeurs de fonction de base du P_1 et de leurs dérivées en x et en y aux points de la formule de quadrature numéro 6 avec l'utilisation de `assemble_coeff_T`.

EXEMPLE 1

```
%Cas de la formule d'integration numero 6 (qui est aussi
% d'ordre 6) et du P1 continu :
clear
X = [0,1,0] ;
Y = [0,0,1] ;
D.intT = 6 ;
[A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, L1, L2, L3] = ...
    assemble_coeff_T(X, Y, D) ;
[Z, DIM] = F_base_T(1, 0, [], [], A1, A2, B1, B2, C1, C2, ...
    DE, D, L1, L2, L3) ;
[Zx, DIM] = F_base_T(1, 1, [], [], A1, A2, B1, B2, C1, C2, ...
    DE, D, L1, L2, L3) ;
[Zy, DIM] = F_base_T(1, 2, [], [], A1, A2, B1, B2, C1, C2, ...
    DE, D, L1, L2, L3) ;
```

Pour les 3 fonctions de base et leurs dérivées partielles d'ordre un, on trouve les valeurs suivantes aux 28 points de la formule de quadrature :

$$Z = \frac{1}{6} \begin{pmatrix} 6 & 5 & 4 & 3 & 2 & 1 & 0 & 5 & 4 & 3 & 2 & 1 & 0 & 4 & 3 & 2 & 1 & 0 & 3 & 2 & 1 & 0 & 2 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 & 4 & 4 & 4 & 5 & 5 & 6 \end{pmatrix}$$

$$Z_x = \begin{pmatrix} -1 & \cdots & -1 \\ 1 & \cdots & 1 \\ 0 & \cdots & 0 \end{pmatrix}; Z_y = \begin{pmatrix} -1 & \cdots & -1 \\ 0 & \cdots & 0 \\ 1 & \cdots & 1 \end{pmatrix}.$$

Exemple 2 : Exemple sans utilisation de `assemble_coeff_T`. Cet exemple traite le cas des tracés des fonctions de base du $P_2(K)$ et de leurs dérivées, avec K triangle de sommets $(0,0)$, $(1,0)$ et $(0,1)$. On utilise le maillage suivant. Tout d'abord on maille le carré unité en n^2 sous-carrés avec `meshgrid` et la valeur de $n = 99$, puis, pour utiliser `surf`, on remplace chaque sommet (x,y) du maillage obtenu placé en dehors de K , par sa projection orthogonale sur la droite $1 - x - y = 0$: les coordonnées de la projection sont alors données par $(x - y + 1)/2$ et $(y - x + 1)/2$.

Cette technique permet ensuite d'utiliser la fonction `surf` sur le triangle. Sinon, on pourra aussi utiliser les fonctions de tracé du code.

Ensuite, on calcule $L1$, $L2$ et $L3$, les matrices carrées de taille $n + 1$, coordonnées barycentriques des sommets maillage obtenu, puis on passe à la fonction `F_base_T` les vecteurs lignes $L1(:)'$, $L2(:)'$, $L3(:)'$, pour le calcul des fonctions de base aux sommets du maillage. Pour une matrice M de taille $n + 1$ la formule qui permet de passer de $M(:)$ à M est donnée par `reshape(., n + 1, n + 1)`, C'est cette formule que l'on applique à $Z(k, :)$ qui est la ligne k de la variable résultat Z , $Z(k, :)$ est alors de même dimension que $L1$ et représente les valeurs aux points du maillage de la k -ième fonction de base. On obtient alors une représentation des 6 fonctions de base du P_2 en appliquant la fonction `surf` à $ZG = \text{reshape}(Z(k, :), n + 1, n + 1)$. Le second exemple du `help` de la fonction `F_base_T` est le suivant.

EXEMPLE 2

```
clear ;
close all ;
%Maillage du carre
X = [0,1,0] ;
Y = [0,0,1] ;
n = 99 ;
XD = (0 :n)/n ;
YD = (0 :n)/n ;
%Calcul d'une grille du triangle :
%1) d'abord avec MESHGRID qui maille le carre unite :
[XG, YG] = meshgrid(XD, YD) ;
%2) Ensuite, les points en dehors du triangle sont remplaces
%par leur projection orthogonale orthogonalement sur la droite
%1 - x - y = 0 :
I = find(1 - XG - YG < 0 - eps) ;
X1 = (XG(I) - YG(I) + 1)/2 ;
Y1 = (YG(I) - XG(I) + 1)/2 ;
XG(I) = X1 ;
YG(I) = Y1 ;
%Coordonnees barycentriques :
L1 = 1 - XG - YG ;
L2 = XG ;
L3 = YG ;
```

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base_T

```
[Z, DIM] = F_base_T(2, 0, [], [], X(1), Y(1), ...
    X(2), Y(2), X(3), Y(3), 2, [], L1( :)', L2( :)', L3( :)' );
%Inversion de ( :)' : Exemple : soit A = reshape(1 :n*n,n,n),
%on revient a A en partant de A( :)' par reshape(A( :)',n,n)
for k=1 :DIM
    ZG = reshape(Z(k, :), n+1, n+1);
    figure(k);
    surf(XG, YG, ZG, ZG);
end;
```

Algorithme :

C :/EF/Assemblage_elementaire/F_base_T.m

On remet ici la partie déclarative de la fonction :

```
function [z, dim] = F_base_T(e, d, A, B, a1, a2, b1, b2, c1, ...
    c2, de, D, L1, L2, L3);
```

L'algorithme consiste à calculer, en fonction de la variable **espace** les variables **deg1** et **deg2** qui permettent de définir l'espace d'EF E_K sur le triangle K comme étant l'espace

$$\mathcal{P}_{i,j}(K) = \mathcal{A}_i(K) \oplus B_j(K),$$

décrit au paragraphe 1.4, avec **deg1** = i et **deg2** = j puis à utiliser avec ces 2 variables la fonction **F_base**.

- Deux familles d'espaces d'EF :
- Famille 1. Cette famille comprend les EF numérotés de 1 à 199 :

```
cas = 1 :199;
```

- Famille 2. Cette famille comprend les EF numérotés 200 et 300 pour le P_{10} continu et discontinu, l'addition de cette famille illustre l'ajout d'une nouvelle famille au code EF dans **F_base_T** :

```
cas2 = [200,300];
```

On détermine d'abord à quel ensemble formé par **cas** et **cas2** appartient **e**, puis les valeurs de **deg1** et **deg2** qui ici définissent complètement ici l'espace d'EF utilisé. On remarque que sur le triangle, espaces d'EF continus et discontinus sont identiques.

Si $e \in \{1, \dots, 199\}$

Si $e == 100$

```
deg1 = 0;
```

```
deg2 = 0;
```

```

        dim = 1 ;
Fin;
Si  $1 \leq e \leq 9$ 
    deg1 = e ;
    deg2 = e ;
    dim = (e + 1)*(e + 2)/2 ;
Fin;
Si  $10 \leq e \leq 99$ 
    deg2 = e - 10.*floor(e./10) ;%mod(e,10)
    deg1 = (e - deg2)/10 ;
    dim = (3*deg1 + (deg2>0)*(deg2 - 1)*(deg2- 2)/2) ;
Fin;
Si  $100 \leq e \leq 109$ 
    eb = e - 100 ;
    deg1 = eb ;
    deg2 = eb ;
    dim = (eb + 1)*(eb + 2)/2 ;
Fin;
Si  $110 \leq e \leq 199$ 
    eb = e -100 ;
    deg2 = eb - 10.*floor(eb./10) ;%mod(e,10)
    deg1 = (eb - deg2)/10 ;
    dim = (3*deg1 + (deg2>0)*(deg2 - 1)*(deg2- 2)/2) ;
Fin;
Fin; (du si  $e \in \{1, \dots, 199\}$ ).
Si  $e \in \{200, 300\}$ 
    deg1 = 10 ;
    deg2 = 10 ;
    dim = 66 ;
Fin;
Si une des variables L1, L2 ou L3 est vide alors
    z = [] ;
Fin;
• Cas de la famille 1 :
```


4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base_A

```
z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, L1, L2, L3, ...  
                                deg1, deg2) ;
```

- Cas de la famille 2 (même appel ici) :

```
z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, L1, L2, L3, ...  
                                deg1, deg2) ;
```

Remarque. Actuellement e n'est pas utilisé dans `F_base`, on peut poser $e = []$.

Fonction :

```
function [z, dim] = F_base_A(e, d, A, B, a1, a2, b1, b2, c1, ...  
                             c2, de, D, L1, L2, L3) ;
```

Actuellement identique à `F_base_T`.

Fonction :

```
function z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, L1, L2, ...
                    L3, deg1, deg2) ;
```

Valeur des fonctions de base d'un espace d'EF E_K sur un triangle K . Les valeurs sont données en des points repérés par leurs coordonnées barycentriques. Ces points sont par exemple ceux issus d'une formule de quadrature sur les triangles.

Remarque. Dans sa version actuelle, `F_base` ne dépend pas de la variable `e` (notée **espace** ci-dessous) qui peut être mise à `[]`.

L'appel :

```
z = F_base(espace, der, a1, a2, b1, b2, c1, c2, de, L1, ...
           L2, L3, deg1, deg2) ;
```

(avec `der` pour `d` dans la déclaration de la fonction) renvoie les valeurs des fonctions de base de l'espace vectoriel $\mathcal{P}_{deg1,deg2}(K)$, avec `deg1 = deg1`, `deg2 = deg2`, $\mathcal{P}_{deg1,deg2}(K)$ étant l'espace vectoriel décrit au Paragraphe 1.4, avec ses `ddl` et ses fonctions de base associées.

Les valeurs obtenues sont alors aussi celles des fonctions de base et de leurs dérivées de l'espace d'EF E_K sur K défini par $E_K = P_{deg1,deg2}^d$ ou $E_K = P_{deg1,deg2}^c$. Ici le résultat est identique dans le cas discontinu ou continu.

Les valeurs des fonctions de base sont données aux points de coordonnées barycentriques relatives au triangle K de sommets $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$ et qui sont données par les vecteurs lignes `L1`, `L2`, `L3`. La dérivation est indiquée par `der = 0`, `1` ou `2` et `de` est le Jacobien donné par :

$$de = [a1, b1, c1] * [b2 - c2, -a2 + c2, -b2 + a2]'$$

En général les points donné par `L1`, `L2` et `L3` sont issus d'une des formules de quadrature utilisée pour l'intégration sur le triangle (ou ses arêtes) de sommets de coordonnées $(a1, a2)$, $(b1, b2)$ et $(c1, c2)$. La fonction est actuellement indépendante de la variable **espace** qui désigne le numéro de l'espace d'EF E_K et la variable, **espace** peut être mise à `[]`.

En sortie, `Z` est la matrice $\dim(E_K) \times \text{length}(L1)$ des valeurs des fonctions de base telle que $Z(i, j)$ est la valeur de la dérivée partielle `der = 0, 1` ou `2` de la i -ème fonction de base de E_K au point de coordonnées barycentriques $L1(j)$, $L2(j)$, $L3(j)$. La variable `Z` vaut `[]` si une des variables `L1`, `L2` ou `L3` vaut `[]`.

Variables d'entrée :

espace : `DOUBLE ARRAY`. Numéro d'espace d'EF sur un triangle K . Ces espaces sont décrits au Paragraphe 1.4. La description de cette variable est donnée à la fonction `F_base_T` ou `pre_assemble_T`, cf. aussi les exemple donnés dans ces deux

4.3. DESCRIPTION DES FONCTIONS FONCTION `Assemblage_elementaire/F_base`

fonctions. La fonction ne dépend pas actuellement de cette variable qui peut être mise à [].

der : DOUBLE ARRAY. Les valeurs de **der** = 0, 1, 2 indiquent respectivement la dérivation à l'ordre 0, par rapport à la première variable x et par rapport à la deuxième variable y .

Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature.

a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K .

de : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

D : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ D du Tableau 2.2.1-1c, Paragraphe 2.2.1.

L1, L2, L3 : DOUBLE ARRAY. Vecteurs lignes de tailles identiques de coordonnées barycentriques relativement au triangle de sommets de coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) .

deg1, deg2 : DOUBLE ARRAY. Scalaires qui permettent de définir l'espace vectoriel $E(K)$ sur le triangle K suivant :

$$E(K) = P_{i,j}(K) = \mathcal{A}_i(K) \oplus \mathcal{B}_j(K),$$

décrit au Paragraphe 1.4 avec ses *ddl* et ses fonctions de base associées, avec **deg1** = i et **deg2** = j . L'espace $E(K)$ est alors l'espace vectoriel associé à l'espace d'EF E_K sur K défini par $E_K = P_{deg1,deg2}^d$ ou $E_K = P_{deg1,deg2}^c$.

Variables de sortie :

Z : DOUBLE ARRAY. Matrice de $\dim(E_K) \times \text{length}(\text{L1})$ des valeurs des fonctions de base d'un espace d'EF E_K sur K , telle que $Z(i, j)$ est la valeur de la dérivée partielle **der** = 0, 1 ou 2 de la i -ème fonction de base de E_K au point de coordonnées barycentriques $\text{L1}(j)$, $\text{L2}(j)$ et $\text{L3}(j)$. La variable **Z** vaut [] si une des variables **L1**, **L2** ou **L3** prend la valeur [].

Variables facultatives : Dans sa version actuelle la fonction ne dépend pas de la variable **espace** qui peut être mise à [].

Exemple : Cet exemple est tiré du `help` de la fonction `F_base` pour le calcul des fonctions de bases du $P_1 \oplus$ bulle P_3 .

EXEMPLE

```
%Valeurs aux points de la formule de quadrature numero 6 pour
%l'integration sur les triangles (qui est aussi d'ordre 6) des
%fonctions de base et de leurs derivees a l'ordre 1 de l'espa-
%ce P1 + bulle P3 continu. Dans l'appel de F_BASE ceci se fait
%en posant DEG1 = 1, DEG2 = 3 et ESPACE = 13 (dans sa version
%actuelle, F_BASE ne depend pas de la valeur de ESPACE).

X = [0,1,0] ;
Y = [0,0,1] ;
D.intT = 6 ;
[A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, L1, L2, L3] = ...
                                assemble_coeff_T(X,Y,D) ;
Z = F_base([], 0, A1, A2, B1, B2, C1, C2, DE, L1, L2, L3,1,3) ;
Zx = F_base([], 1, A1, A2, B1, B2, C1, C2, DE, L1, L2, L3,1,3) ;
Zy = F_base([], 2, A1, A2, B1, B2, C1, C2, DE, L1, L2, L3,1,3) ;
```

Algorithme :

EF/Assemblage_elementaire/F_base.m

On remet ici la partie déclarative de la fonction :

```
function z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, L1, L2, ...
                                L3, deg1, deg2) ;
```

L'algorithme est basé sur le Paragraphe 2.3.3 qui décrit l'ordre choisi pour les *ddl* de l'espace

$$P_{i,j}(K) = \mathcal{A}_i(K) \oplus B_j(K),$$

défini au Paragraphe 1.4 et donne une formule pour le calcul des fonctions de base.

On remet ici la partie déclarative de la fonction :

```
function z = F_base(e, d, a1, a2, b1, b2, c1, c2, de, L1, L2, ...
                                L3, deg1, deg2) ;
```

On note $a_1 = (a1, a2)$, $a_2 = (b1, b2)$, $a_3 = (c1, c2)$ les sommets du triangle K et ℓ_1 , ℓ_2 et ℓ_3 les coordonnées barycentriques relatives à ces sommets. Dans cette fonction les variables `deg1` et `deg2` représentent les valeurs de i et j de l'espace $P_{i,j}(K)$.

4.3. DESCRIPTION DES FONCTIONS FONCTION `Assemblage_elementaire/F_base`

Nous utiliserons les *ddl* du $P_k(K)$ donnés pour $k \geq 1$ par

$$a_{pqr} = \frac{p}{k}a_1 + \frac{q}{k}a_2 + \frac{r}{k}a_3, \forall (p, q, n) \in \mathbb{N}^2, p + q + r = k$$

et par a_{000} le centre de gravité de K pour le $P_0(K)$. Ces *ddl* seront décrits dans l'ordre donné au Paragraphe 2.3.3.

Si d est un nombre différent de 0, 1 ou 2 alors un message d'erreur s'affiche et la fonction stoppe le programme avec un message d'erreur.

Si une des variables `L1`, `L2`, `L3` est vide alors on sort de la fonction avec la valeur $z = []$.

On pose `nb` le nombre de points où l'on calcule les valeurs des fonctions de base. Ce sera le nombre de colonnes de la variable de sortie z , le nombre de lignes étant la dimension de l'espace $P_{deg1,deg2}(K)$:

`nb = length(L1) ;`

- Si `deg1 + deg2 = 0`. On est dans le cas du $P_0(K)$ et suivant la valeur de `d` la variable de dérivation on a alors

cas `d = 0` :

`z = ones(1,nb) ;`

cas `d = 1` :

`z = zeros(1,nb) ;`

cas `d = 2` :

`z = zeros(1,nb) ;`

et on sort de la fonction.

- Si `deg1 + deg2 ≠ 0`.

- Calcul des dérivées des coordonnées barycentriques si `d = 1` ou bien `d = 2`. Les dérivées s'expriment en fonction des sommets (a_1, a_2) , (b_1, b_2) , (c_1, c_2) du triangle courant. Leur expression est donnée au Paragraphe 2.3.3. On pose ci-dessous `d12 = 0` si `d = 0` et `d12 = 1` si `d = 1` ou `2`. Les éléments mathématiques de ce calcul sont décrits au Paragraphe 3.4.

`d12 = 0 ;`

Si `d==1`

`d12 = 1 ;`

`t2 = 1/de.*ones(1,nb) ;`

`L1_d = (b2-c2)*t2 ;`

`L2_d = -(a2-c2)*t2 ;`

`L3_d = (-b2+a2)*t2 ;`

Sinon, si `d==2`

```

d12 = 1 ;
t2 = 1/de.*ones(1,nb) ;
L1_d = (c1-b1)*t2 ;
L2_d = -(c1-a1)*t2 ;
L3_d = (b1-a1)*t2 ;
Fin ;

```

- Ensuite, on passe au calcul des fonctions de base des espace $\mathcal{A}_{deg1}(K)$ et $B_{deg2}(K)$: on calcule la valeur des ces fonctions aux points de $\mathbb{R}^2$ donnés par leur coordonnées barycentriques L1, L2 et L3.

1) Calcul de la base de l'espace $\mathcal{A}_{deg1}(K)$.

La base de $\mathcal{A}_{deg1}(K)$ est la base du $P_{deg1}(K)$ privé des fonctions attachées à un *ddl* intérieur.

- Pour accélérer dans la fonction les calculs, les cas `deg1 = 1` ou `deg1 = 2` sont traités à part. Ceci est fait écrivant les formules de calcul en extension, sans boucles. Il est possible d'utiliser cette écriture pour d'autres éléments dont on voudrait accélérer la vitesse de calcul. Les éléments mathématiques de ce calcul sont décrits au Paragraphe 3.4, pour ces deux valeurs du degré.

```

Si deg1==1
    Si d==0
        z = [L1 ;L2 ;L3] ;
    sinon
        z = [L1_d ;L2_d ;L3_d] ;
    Fin ;
Fin ;
Si deg1==2
    Si d==0
        L11 = 2*L1.*(L1 - 1/2) ;
        L22 = 2*L2.*(L2 - 1/2) ;
        L33 = 2*L3.*(L3 - 1/2) ;
        L12 = 4*L1.*L2 ;
        L13 = 4*L1.*L3 ;
        L23 = 4*L2.*L3 ;
        z = [L11 ;L22 ;L33 ;L12 ;L13 ;L23] ;
    Sinon, si d12
        L11_d = L1_d.*(4*L1 - 1) ;

```

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base

```

L22_d = L2_d.*(4*L2 - 1) ;
L33_d = L3_d.*(4*L3 - 1) ;
L12_d = 4*L1_d.*L2 + 4*L1.*L2_d ;
L13_d = 4*L1_d.*L3 + 4*L1.*L3_d ;
L23_d = 4*L2_d.*L3 + 4*L2.*L3_d ;
z = [L11_d ; L22_d ; L33_d ; L12_d ; L13_d ; L23_d] ;
Fin ;
Fin ;

```

- Traitement du cas $\text{deg1} > 2$. Remarque : l'algorithme ci-dessous fonctionne aussi pour les cas $\text{deg1} = 1$ ou 2 déjà traités ci-dessus.

Si $\text{deg1} > 2$:

- Allocation mémoire :

```

Lp1 = zeros(deg1 + 1, length(L1)) ;
Lp2 = Lp1 ;
Lp3 = Lp1 ;

```

- Allocation mémoire pour les dérivées si $\text{d12} = 1$:

```

Si d12
    Lp1_d = Lp1 ;
    Lp2_d = Lp1 ;
    Lp3_d = Lp1 ;
Fin ;

```

- Calcul des fonctions $L_j^{k,i}$, $i = 1, 2, 3$ définies au Paragraphe 2.3.3 avec $k = \text{deg1}$. Les formules du Paragraphe 2.3.3, donnent pour $1 \leq i \leq 3$ (en changeant dans l'expression $L_j^{k,i}$ du Paragraphe 2.3.3, j en k et k en deg1 ce qui donne $L_k^{\text{deg1},i}$), on a :

$$L_0^{\text{deg1},i} = 1,$$

$$L_k^{\text{deg1},i} = L_{k-1}^{\text{deg1},i}(\ell_i - \frac{k-1}{\text{deg1}}), \quad k = 1, \dots, \text{deg1},$$

et pour les dérivées, avec $d = x$ ou y :

$$L_{0,d}^{\text{deg1},i} = 0,$$

$$L_{k,d}^{\text{deg1},i} = L_{k-1,d}^{\text{deg1},i}(\ell_i - \frac{k-1}{\text{deg1}}) + L_{k-1}^{\text{deg1},i} \ell_{i,d}, \quad k = 1, \dots, \text{deg1},$$

Dans l'algorithme, on place les résultats obtenus pour une valeur k donnée dans $\text{Lp1}(k+1, :)$, $\text{Lp2}(k+1, :)$, $\text{Lp3}(k+1, :)$ et, le cas échéant, pour les dérivées, dans $\text{Lp1\_d}(k+1, :)$, $\text{Lp2\_d}(k+1, :)$, $\text{Lp3\_d}(k+1, :)$, avec $k = 0, \dots, \text{deg1}$.

- Valeurs de $L_0^{deg1,1}$, $L_0^{deg1,2}$, $L_0^{deg1,3}$:

```

Lp1(1, :) = ones(1,length(L1));
Lp2(1, :) = ones(1,length(L1));
Lp3(1, :) = ones(1,length(L1));

```

- Calculs de $L_k^{deg1,1}$, $L_k^{deg1,2}$, $L_k^{deg1,3}$, $k = 1, \dots, \text{deg1}$:

```

Pour k=1 :deg1
    Lp1(k + 1, :) = Lp1(k, :).*(L1 - (k - 1)/deg1);
    Lp2(k + 1, :) = Lp2(k, :).*(L2 - (k - 1)/deg1);
    Lp3(k + 1, :) = Lp3(k, :).*(L3 - (k - 1)/deg1);
    Si d12
        Lp1_d(k + 1, :) = Lp1_d(k, :).*(L1 - (k - 1)/deg1) ...
                        + Lp1(k, :).*L1_d;
        Lp2_d(k + 1, :) = Lp2_d(k, :).*(L2 - (k - 1)/deg1) ...
                        + Lp2(k, :).*L2_d;
        Lp3_d(k + 1, :) = Lp3_d(k, :).*(L3 - (k - 1)/deg1) ...
                        + Lp3(k, :).*L3_d;
    Fin;
Fin;

```

- Coefficient intervenant dans la formule donnant les fonctions de base :

```

a = deg1^deg1;

```

- Calcul des valeurs des fonctions de base aux points désignés par leurs coordonnées barycentriques L1, L2 et L3 et rangement dans un tableau suivant l'ordre décrit au Paragraphe 2.3.3 : fonctions de base associées au 3 sommets, puis successivement, fonctions de base associées aux *ddl* situés sur l'arête $]a, b[$ (extrémités non comptées), l'arête $]a, c[$ et enfin l'arête $]b, c[$.

La fonction de base ψ_{pqr} associée au *ddl* a_{pqr} et les dérivées de cette fonction se calculent avec la formule ci-dessous où l'on a posé $k = \text{deg1}$:

$$\psi_{pqr} = \frac{k^k}{p!q!r!} L_p^{k,1} L_q^{k,2} L_r^{k,3}.$$

$$\psi_{pqr,d} = \frac{k^k}{p!q!r!} (L_{p,d}^{k,1} L_q^{k,2} L_r^{k,3} + L_p^{k,1} L_{q,d}^{k,2} L_r^{k,3} + L_p^{k,1} L_q^{k,2} L_{r,d}^{k,3}), \quad d = 1, 2.$$

- Calcul des fonctions de base associées aux 3 sommets. On calcule les valeurs des fonctions de base associées aux *ddl* sommets de triangle. L'ordre des sommets choisi est (*cf.* Paragraphe 2.3.3) :

$$a_1 = a_{deg100}, \quad a_2 = a_{0deg10}, \quad a_3 = a_{00deg1}$$

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base

de fonctions de base associées

$$\psi_{deg100}, \psi_{0deg10}, \psi_{00deg1}$$

dont les valeurs sont données respectivement par

$$\psi_{deg100} = \frac{deg1^{deg1}}{deg1!} L_{deg1}^{deg1,1}, \psi_{0deg10} = \frac{deg1^{deg1}}{deg1!} L_{deg1}^{deg1,2}, \psi_{00deg1} = \frac{deg1^{deg1}}{deg1!} L_{deg1}^{deg1,3}.$$

Pour ce calcul l'algorithme s'écrit :

```

z = zeros(round(3*deg1),length(L1)) ;
Si d==0
    z(1, :) = a/prod(1 :deg1)*Lp1(end, :) ;
    z(2, :) = a/prod(1 :deg1)*Lp2(end, :) ;
    z(3, :) = a/prod(1 :deg1)*Lp3(end, :) ;
Sinon, si d12
    z(1, :) = a/prod(1 :deg1)*Lp1_d(end, :) ;
    z(2, :) = a/prod(1 :deg1)*Lp2_d(end, :) ;
    z(3, :) = a/prod(1 :deg1)*Lp3_d(end, :) ;
Fin

```

- Calcul des fonctions de base pour l'arête [a1, a2]. On calcule les valeurs des fonctions de base associées aux *ddl* situés sur de l'arête [a1, a2] L'ordre des *ddl* choisi est (cf. Paragraphe 2.3.3) :

$$a_{i,j,0}, i = deg1 - 1, \dots, 1 \text{ avec } j = deg1 - i$$

de fonctions de base associées

$$\psi_{ij0} = \frac{deg1^{deg1}}{i!j!} L_i^{deg1,1} L_j^{deg1,2}.$$

La variable **compt** ci-dessous permet de numéroté dans l'ordre décrit ci-dessus les fonctions de base ayant un *ddl* sur l'arête [a1, a2]. La fonction de base de numéro **compt** est alors placée sur la ligne **z(3 + compt, :)**, les 3 premières lignes de **z** contenant déjà les valeurs des fonctions de base attachée à un sommet du triangle.

```

compt = 0 ;
Pour i = deg1 - 1 :-1 :1,
    j = deg1 - i ;
    p1 = Lp1(1+i, :) ;
    p2 = Lp2(1+j, :) ;

```

```

Si d12
    p1_d = Lp1_d(1+i, :) ;
    p2_d = Lp2_d(1+j, :) ;
Fin ;
f1 = a/(prod(1 :i)) ;
compt = compt + 1 ;
Si d==0
    z(3 + compt, :) = f1/(prod(1 :j))*p1.*p2 ;
Sinon, si d12
    z(3 + compt, :) = f1/(prod(1 :j))*(p1_d.*p2+p1.*p2_d) ;
Fin ;
Fin ;

```

Pour les deux autres arêtes l'algorithme est similaire, il est le suivant.

- Calcul des fonctions de base pour l'arête $[a_1, a_3]$. L'algorithme s'obtient en remplaçant dans l'algorithme précédent Lp2 par Lp3 et Lp2_d par Lp3_d :

```

Pour i = deg1 - 1 :-1 :1,
    j = deg1 - i ;
    p1 = Lp1(1+i, :) ;
    p2 = Lp3(1+j, :) ;
    Si d12
        p1_d = Lp1_d(1+i, :) ;
        p2_d = Lp3_d(1+j, :) ;
    Fin ;
    f1 = a/(prod(1 :i)) ;
    compt = compt + 1 ;
    Si d==0
        z(3 + compt, :) = f1/(prod(1 :j))*p1.*p2 ;
    Sinon, si d12
        z(3 + compt, :) = f1/(prod(1 :j))*(p1_d.*p2+p1.*p2_d) ;
    Fin ;
Fin

```

- Calcul des fonctions de base pour l'arête $[a_2, a_3]$. L'algorithme s'obtient en remplaçant dans l'algorithme précédent Lp1 par Lp2 et Lp1_d par Lp2_d :

```

Pour i = deg1 - 1 :-1 :1,

```

4.3. DESCRIPTION DES FONCTIONS `FONCTION Assemblage_elementaire/F_base`

```

j = deg1 - i ;
p1 = Lp2(1+i, :) ;
p2 = Lp3(1+j, :) ;
Si d12
    p1_d = Lp2_d(1+i, :) ;
    p2_d = Lp3_d(1+j, :) ;
Fin ;
f1 = a/(prod(1:i)) ;
compt = compt + 1 ;
Si d==0
    z(3 + compt, :) = f1/(prod(1:j))*p1.*p2 ;
Sinon, si d12
    z(3 + compt, :) = f1/(prod(1:j))*(p1_d.*p2+p1.*p2_d) ;
Fin ;
Fin

```

- Fin du traitement du cas $\text{deg1} > 2$.

2) Calcul de la base de l'espace $B_{\text{deg2}}(K)$.

- Traitement du cas $\text{deg2} > 3$. On effectue le calcul si $\text{deg2} > 3$, sinon $B_{\text{deg2}}(K)$ est vide et la fonction `F_base` se termine.

• Comme précédemment pour $A_{\text{deg1}}(K)$ on calcule d'abord les valeurs des fonctions $L_j^{\text{deg2},i}$, $i = 1, 2, 3$, $j = 1, \dots, \text{deg2}$ ainsi que, le cas échéant, les valeurs des dérivées. L'algorithme s'obtient en changeant deg1 par deg2 dans l'algorithme du cas $A_{\text{deg1}}(K)$:

```

Si deg2 >= 3
    Lp1 = zeros(deg2 + 1, length(L1)) ;
    Lp2 = Lp1 ;
    Lp3 = Lp1 ;
    Si d12
        Lp1_d = Lp1 ;
        Lp2_d = Lp1 ;
        Lp3_d = Lp1 ;
    Fin ;
    Lp1(1, :) = ones(1, length(L1)) ;
    Lp2(1, :) = ones(1, length(L1)) ;

```

```

    Lp3(1, :) = ones(1,length(L1));
Pour k=1 :deg2
    Lp1(k + 1, :) = Lp1(k, :).*(L1 - (k - 1)/deg2);
    Lp2(k + 1, :) = Lp2(k, :).*(L2 - (k - 1)/deg2);
    Lp3(k + 1, :) = Lp3(k, :).*(L3 - (k - 1)/deg2);
Si d12
    Lp1_d(k + 1, :) = Lp1_d(k, :).*(L1 - (k - 1)/deg2) ...
                        + Lp1(k, :).*L1_d;
    Lp2_d(k + 1, :) = Lp2_d(k, :).*(L2 - (k - 1)/deg2) ...
                        + Lp2(k, :).*L2_d;
    Lp3_d(k + 1, :) = Lp3_d(k, :).*(L3 - (k - 1)/deg2) ...
                        + Lp3(k, :).*L3_d;

Fin;
Fin;

```

- Calcul des fonctions de base associées aux *ddl* situés à l'intérieur du triangles dans l'ordre décrit au Paragraphe 2.3.3.

L'ordre des *ddl* choisi est le suivant (cf. Paragraphe 2.3.3) avec $k = \text{deg2}$:

$$a_{p,k-p-r,r}, r = 1, \dots, k-2, p = k-r-1, \dots, 1$$

de fonctions de base associées ψ_{pqr} dont nous rappelons l'expression utilisée pour leur calcul :

$$\psi_{pqr} = \frac{\text{deg1}^k}{p!q!r!} L_p^{k,1} L_q^{k,2} L_r^{k,3}.$$

et le calcul de leurs dérivées

$$\psi_{pqr,d} = \frac{k^k}{p!q!r!} (L_{p,d}^{k,1} L_q^{k,2} L_r^{k,3} + L_p^{k,1} L_{q,d}^{k,2} L_r^3 + L_p^{k,1} L_q^{k,2r} L_{r,d}^{k,3}), \quad d = 1, 2.$$

La suite de l'algorithme s'écrit :

- La variable **compt** ci-dessous permet de numérotter dans l'ordre décrit ci-dessus les fonctions de base ayant un *ddl* intérieur.

```
compt = 0 ;
```

- La fonction de base de numéro **compt** est alors placée sur la ligne **L_int(compt,:)** de la variable intermédiaire **L_int** ci-dessous. Cette variable sera ensuite concaténée à la fin avec la variable **z** en la plaçant à la suite de **z**.

```
Lp_int = [] ;
```

```
a = deg2^deg2 ;
```

4.3. DESCRIPTION DES FONCTIONS FONCTION Assemblage_elementaire/F_base

```

Lp_int = zeros( round( (deg2 - 2)*(deg2 - 1)/2 ),length(L1)) ;
Pour k = 1 :deg2 - 2,
    Pour i = deg2 - k - 1 :-1 :1,
        j = deg2 - k - i ;
        p1 = Lp1(1+i, :) ;%limiter le cout d'extraction
    Si d12
        p1_d = Lp1_d(1+i, :) ;
    Fin ;
    f1 = a/(prod(1 :i)) ;
    compt = compt + 1 ;

```

- Calcul des valeurs de la fonction de base proprement dit. Ce calcul n'est réalisée que si la condition sur $\text{all}([i,j,k])$ est vraie, c'est à dire si on est sur un ddl intérieur (le $ddl a_{pqr}$ est intérieur si et seulement si $\min(p, q, r) > 0$, cf. l'espace L_k^{int} du Paragraphe 2.3.3). Cette sécurité ne sert pas dans cette version : la condition est toujours satisfaite.

```

    Si all([i,j,k])
        Si d==0
            Lp_int(compt, :) = ...
                f1/(prod(1 :j)*prod(1 :k))*...
                p1.*Lp2( 1 + j , :).*Lp3( 1 + k , :) ;
        Fin ;
        Si d12
            p2 = Lp2( 1 + j , :) ;
            p3 = Lp3( 1 + k , :) ;
            Lp_int(compt, :) = ...
                f1/(prod(1 :j)*prod(1 :k))*(p1_d.*p2.*p3 ...
                + p1.*(Lp2_d(1+j, :).*p3 + p2.*Lp3_d(1+k, :))) ;
        Fin ;
    Fin ;
Fin ;

```

- Réunion des fonctions des deux espaces $\mathcal{A}_i(K)$ et $B_j(K)$:

```
z = [z ; Lp_int] ;
```

- Fin du traitement du cas $\text{deg2} > 3$, fin du cas $\text{deg1} + \text{deg2} \neq 0$ et fin de la fonction.

Fonction :

```
function [a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3]...
                                         = assemble_coeff_T(X, Y, D) ;
```

Formules de quadrature sur un triangle.

```
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
                                         assemble_coeff_T(X, Y, D) ;
```

renvoie les coefficients et les coordonnées des points de la formule de quadrature numéro `D.intT` sur le triangle K de sommets donnés par les vecteurs lignes ou colonnes `X` et `Y`. Les formules de quadrature actuellement programmées ont pour numéro 4, 5, 6, 8 ou 10 qui est aussi l'ordre de la formule d'intégration.

En sortie

$$(a1, a2) = (X(1), Y(1)), (b1, b2) = (X(2), Y(2)), (c1, c2) = (X(3), Y(3))$$

sont les coordonnées des sommets a, b, c du triangle K et la variable `de` prend la valeur du Jacobien de la bijection affine du triangle de référence $(0, 0), (1, 0), (0, 1)$ sur le triangle K qui est :

$$de = [a1, b1, c1] * [b2 - c2, -a2 + c2, -b2 + a2]'$$

Les variables `Ac` et `Bc` sont les vecteurs lignes des coordonnées cartésiennes en x et en y des points de la formule de quadrature et `L1, L2` et `L3` les vecteurs lignes des coordonnées barycentriques des points de la formule de quadrature. Enfin `coeff` est le vecteur ligne des coefficients de la formule de quadrature.

L'intégrale sur K d'une fonction f est alors approchée par la somme

$$\sum_{i=1}^n \text{coeff}(i) f(\text{Ac}(i), \text{Bc}(i))$$

avec $n = \text{length}(\text{coeff})$, nombre de points de la formule de quadrature.

Par exemple, si la fonction f est représentée par une variable `f` de type `inline`, alors la somme ci-dessus se calcule par

$$\text{Int} = \text{sum}(\text{coeff} * f(\text{Ac}, \text{Bc})).$$

Variables d'entrée :

`X, Y : DOUBLE ARRAY`. Vecteurs lignes ou colonnes telles que $(X(1), Y(1)), (X(2), Y(2)), (X(3), Y(3))$ sont les coordonnées des trois sommets du triangle courant. On note K ce triangle sur lequel on calcule les formules de quadrature.

`D : STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1.

Le champ `D.intT` donne le numéro de la formule d'intégration. Les autres champs de `D` sont facultatifs et non utilisés dans la fonction. Dans la version actuelle de la fonction `D.intT` doit prendre une valeur de l'ensemble $\{4, 5, 6, 8, 10\}$.

Variables de sortie :

a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K . Ici ces variables sont les coordonnées du triangle K courant : on a la relation $\mathbf{X} = [\mathbf{a1}, \mathbf{b1}, \mathbf{c1}]$ et $\mathbf{Y} = [\mathbf{a2}, \mathbf{b2}, \mathbf{c2}]$ si $\mathbf{X}$ et $\mathbf{Y}$ sont des vecteurs lignes ou en transposant dans le cas de vecteurs colonnes.

de : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature.

coeff : DOUBLE ARRAY. Vecteur ligne des coefficients de la formule de quadrature.

L1, L2, L3 : DOUBLE ARRAY. Vecteurs ligne des coordonnées barycentriques des points de la formule de quadrature.

Les exemples ci-dessous sont tirés du **help** de la fonction **assemble_coeff_T**.

Exemple 1 : Dans cet exemple on considère un triangle de sommets $a = (0, 0)$, $b = (1, 0)$, $c = (0, 1)$. La formule de quadrature est celle d'ordre 4. On calcule la surface de ce triangle ainsi que le volume du tétraèdre formé du triangle et du sommet $(0, 0, 1)$. D'autre part on trace sur une figure les points de la formule de quadrature ainsi que leur ordre de numérotation.

EXEMPLE 1 :

```
%Integration de deux fonctions et traces des points de la for-
%
%mule de quadrature utilisee pour approcher les integrales :
clear
close all
f = inline('x.^0','x','y');
g = inline('1 - x - y','x','y');
X = [0,1,0];
Y = [0,0,1];
D.intT = 4;
[A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, L1, L2, L3] = ...
```

```

assemble_coeff_T(X,Y,D) ;

disp('Surface du triangle')
If = sum(COEFF.*f(Ac, Bc))
disp('Volume du tetraedre') ;
Ig = sum(COEFF.*g(Ac, Bc))
%Trace du triangle et des points :
hold on
patch(X,Y,'w') ;
for i=1 :3,
    text(X(i) - 0.09,Y(i) - 0.05 + 0.15*(i==3), ...
        ['Sommet n ', num2str(i)], 'FontSize',16) ;
end ;
for i=1 :length(Ac),
    text(Ac(i) + 0.02, Bc(i) + 0.02, num2str(i), 'FontSize',16) ;
end ;
plot(Ac,Bc,'ko') ;
title(['Points de la formule de quadrature dans leur' ...
        ' ordre de numerotation']);

axis([-0.1,1.2,-0.1,1.2]) ;
%Eventuelle sauvegarde de la figure obtenue : enlever le %
%print('-dpng','C :/EF_text/figures/points_quadrature.png') ;

```

Les 15 points de la formule de quadrature de cet exemple sont donnés à la figure 4.3-8.

Points de la formule de quadrature dans leur ordre de numérotation

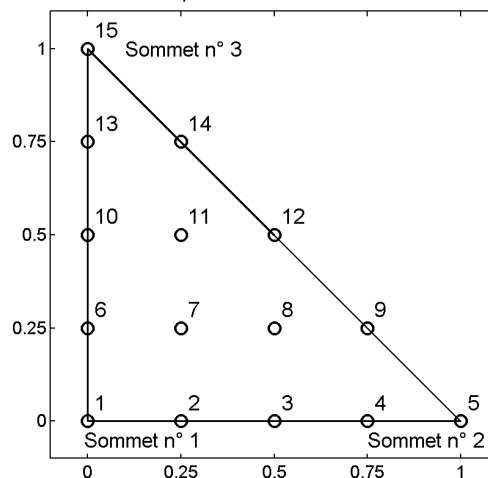


Figure 4.3-8. Points de la formule de quadrature numéro 4 sur le triangle.

Exemple 2 : On teste l'ordre des méthodes dans cet exemple. Soit K le triangle de sommets $(0,0)$, $(1/2,0)$, $(0,1)$ de fonctions de base de Lagrange données par $2x$, y , $1 - 2x - y$, On a la relation :

$$\int_K (2x)^i y^j = 2|K| \frac{i!j!}{(i+j+2)!}.$$

Une formule de quadrature d'ordre n sur K doit redonner cette valeur pour tout couple (i,j) d'entiers tel que $i+j \leq n$. Ci-dessous on teste les méthodes de numéros $m = 4, 5, 6, 10$. Pour chaque valeur de m on affiche la valeur `disp(max(max(abs(L))), L(i,j), 1 ≤ i ≤ m, 0 ≤ j ≤ m - i)` donnant l'erreur d'intégration $\int_K (2x)^i y^j$ qui est ici nulle, à l'erreur machine près, pour ces valeurs de m , i et j .

Le second exemple du `help` de la fonction `assemble_coeff_T` et donné par :

EXEMPLE 2 :

```
%Test de verification de l'ordre des formules d'integration :
%Ces formules doivent etre exactes pour (2*x)^i*y^j,
%pour tout couple d'entier (i,j) tel que i + j <= ordre
%et l'erreur L ci-dessous doit etre nul a la precision
%machine pres.

clear
for m = [4,5,6,8,10] ; %On teste chacune des methodes
    L = [] ;
    D.intT = m ;
    n = m ; %ordre et numero de formule sont identiques
    X = [0,1/2,0] ;
    Y = [0,0,1] ;
    [A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, ...
      L1, L2, L3] = assemble_coeff_T(X,Y,D) ;
    for i=0 :n
        for j=0 :n-i
            %Tests avec sur ce triangle (2*x)^i*y^j, 2*x, et y
            %sont deux des trois fonctions de base de Lagrange :
            f = inline(['(2*x).^',num2str(i), ...
                        '.*y.^',num2str(j)], 'x', 'y') ;
            L(i+1,j+1) = sum(COEFF.*f(Ac, Bc)) - ...
            abs(DE)*factorial(i)*factorial(j)/factorial(i + j + 2) ;
```

```

        end ;
    end ;
    disp(max(max(abs(L))))
end ;

```

Algorithme :

EF/Assemblage_elementaire/assemble_coeff_T.m

On remet ici la partie déclarative de la fonction :

```

function [a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3]...
    = assemble_coeff_T(X, Y, D) ;

```

Le principe du calcul des formules de quadrature est donné au Paragraphe 3.3.1 pour la formule d'ordre 4. Les coefficients des formules sont calculés dans les fichiers .mws indiqués au Paragraphe 3.3.1.

On reprend ci-dessous, le principe de calcul, en reprenant notamment l'ordre des points de la formule de quadrature.

Soient a_1, a_2, a_3 les sommets d'un triangle K et ℓ_1, ℓ_2 et ℓ_3 les coordonnées barycentriques relatives à ces trois sommets. Pour la formule d'ordre n les points de la formule de quadrature sont donnés par :

$$a_{ijk} = \frac{i}{n}a_1 + \frac{j}{n}a_2 + \frac{k}{n}a_3, \text{ avec } (i, j, k) \in \mathbb{N}^3, \text{ tel que } 0 \leq i + j \leq n, i + j + k = n.$$

On a donc $\ell_1(a_{ijk}) = i, \ell_2(a_{ijk}) = j, \ell_3(a_{ijk}) = k$.

Les points sont alors choisis dans l'ordre suivant :

- on pose $k = 0$ et i décroît de n à 0 (points de la droite $[a_1, a_2]$) :

$$a_{n,0,0}, a_{n-1,1,0}, a_{n-2,2,0}, \dots, a_{n-j,j,0}, \dots, a_{0,n,0},$$

- ensuite on pose $k = 1$ et i décroît de $n - 1$ à 0 (points de la droite parallèle à $[a_1, a_2]$ passant par $a_1 + \frac{1}{n}a_3$) :

$$a_{n-1,0,1}, a_{n-2,1,1}, a_{n-3,2,1}, \dots, a_{n-1-j,j,1}, \dots, a_{0,n-1,1},$$

- puis pour k quelconque et i décroissant de $n - k$ à 0 (points de la droite parallèle à $[a_1, a_2]$ passant par $a_1 + \frac{k}{n}a_3$) :

$$a_{n-k,0,k}, a_{n-k-1,1,k}, a_{n-k-2,2,k}, \dots, a_{n-k-j,j,k}, \dots, a_{0,n-k,k},$$

et enfin $k = n$ (point a_3) :

$$a_{0,0,n}.$$

Notons $\alpha = (n+1)(n+2)/2$ le nombre de points a_{ijk} définis ci-dessus et $(m_p)_{1 \leq p \leq \alpha}$ le vecteur des points a_{ijk} rangés dans l'ordre ci-dessus.

Dans la fonction `assemble_coeff_T`, $Ac(p)$ et $Bc(p)$ sont alors les coordonnées cartésiennes du point m_p et $L1(p)$, $L2(p)$ et $L3(p)$ ses coordonnées barycentriques données respectivement par $\ell_1(m_p)$, $\ell_2(m_p)$, $\ell_3(m_p)$. On peut visualiser l'ordre de ces points d'intégration avec la Figure 4.3-8 de l'exemple 1 ci-dessus obtenue pour $n = 4$. Le vecteur `coeff(p)` est alors le coefficient de la formule de quadrature associé au point m_p . Ces coefficients sont donnés dans la fonction `assemble_coeff_T`.

Nous donnons ci-dessous une manière de calculer les coefficients de la formule de quadrature. On note $(c_p)_p$ le vecteur des coefficients associé m_p dans la formule de quadrature pour approcher l'intégrale $\int_K f(x, y) dx dy$ i.e.

$$\int_K f(x, y) dx dy \simeq \sum_{p=1}^{\alpha} c_p f(m_p).$$

On rappelle que pour un triangle K on a noté $|K|$ sa surface. Pour que la formule de quadrature soit au moins d'ordre n , il suffit que les coefficients c_p vérifient pour $0 \leq r + s \leq n$:

$$\sum_{p=1}^{\alpha} c_p \ell_1(m_p)^r \ell_2(m_p)^s = \int_K \ell_1^r \ell_2^s = 2|K| \frac{r!s!}{(2+r+s)!},$$

car la famille $\{\ell_1^r \ell_2^s\}_{0 \leq r+s \leq n}$ est une base de $P_n(K)$.

Pour écrire le système matriciel vérifié par c , on choisit de décrire l'ensemble $\{0 \leq r + s \leq n\}$ qui contient $(n+1)(n+2)/2$ termes avec l'ordre suivant. On pose pour r le vecteur qui contient les entiers allant de n à 0 , puis les entiers allant de $n-1$ à 0 , etc. puis les entiers allant de $n-j$ à 0 , etc. puis les entiers 1 et 0 , puis 0 :

$$r = (n, n-1, \dots, 1, 0, n-1, \dots, 0, \dots, n-j, \dots, 0, \dots, 1, 0, 0),$$

On pose ensuite pour s le vecteur :

$$s = (0, \dots, 0, 1, \dots, 1, \dots, k, \dots, k, \dots, n-1, n-1, n),$$

où le terme 0 figure n fois, le terme 1 figure $n-1$ fois, le terme k figure $n-k+1$ fois etc. et le terme n figure 1 fois.

L'ensemble $\{0 \leq r + s \leq n\}$ est alors décrit par l'ensemble des couples $\{(r(j), s(j))\}_{1 \leq j \leq \alpha}$.

On remarque que r est aussi le vecteur des valeurs successives prises par le premier indice de a_{ijk} lorsque l'on décrit l'ensemble des points a_{ijk} dans l'ordre ci-dessus. On obtient la même chose pour s mais relativement au troisième indice.

Remarque. Avec ce choix on a donc la relation $r(p) = n\ell_1(m_p)$ et $s(p) = n\ell_3(m_p)$.

Posons

$$\hat{c} = \frac{c}{2|K|}.$$

Le système précédent s'écrit alors de manière équivalente

$$\sum_{p=1}^{\alpha} \hat{c}_p \ell_1(m_p)^{r(j)} \ell_2(m_p)^{s(j)} = \frac{1}{2|K|} \int_K \ell_1^{r(j)} \ell_2^{s(j)} = \frac{r(j)!s(j)!}{(2+r(j)+s(j))!},$$

pour tout $1 \leq j \leq (n+1)(n+2)/2$. Ce qui donne c solution de

$$A\hat{c} = f, c = 2|K|\hat{c}$$

avec

$$A_{jp} = \ell_1(m_p)^{r(j)} \ell_2(m_p)^{s(j)}, f(j) = \frac{r(j)!s(j)!}{(2+r(j)+s(j))!}.$$

On remarque que les matrices A et f , et donc $\hat{c}$, ne dépendent pas du triangle K . Lorsque le triangle K est le triangle de référence $\hat{K}$ de sommets $(0,0)$, $(1,0)$, $(0,1)$, comme sa surface est $1/2$ alors $\hat{c} = c$ et $\hat{c}$ est donc le vecteur des coefficients de la formule de quadrature pour ce triangle.

C'est le système $A\hat{c} = f$, qui est calculé et résolu dans le fichier de calcul formel :

EF/z_info_quadratures/quadratures_T.mws.

Dans ce fichier on calcule d'abord les coordonnées barycentriques L1, L2, L3 de valeurs respectives $\ell_1(m_p)$, $\ell_2(m_p)$, $\ell_3(m_p)$, les valeurs de r et s étant représentées dans les calculs par $n * L1 = r$ et $n * L3 = s$ (cf. la remarque précédente). Dans ce fichier la valeur de n est à choisir suivant la formule de quadrature que l'on veut calculer (n vaut 4 dans le fichier initial).

Ci-dessous nous avons mis l'équivalent de ce fichier en version .m, pour la valeur de $n = 4$. Dans ce fichier on calcule la valeur des coordonnées barycentriques des points de la formule de quadrature :

$$\begin{aligned} L1 &= (4, 3, 2, 1, 0, 3, 2, 1, 0, 2, 1, 0, 1, 0, 0)/4, \\ L2 &= (0, 1, 2, 3, 4, 0, 1, 2, 3, 0, 1, 2, 0, 1, 0)/4, \\ L3 &= (0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 3, 3, 4)/4 \end{aligned}$$

puis la matrice A et le second membre f du système dont la résolution donne la valeur des coefficients de la formule de quadrature :

$$C = (0, 4, -1, 4, 0, 4, 8, 8, 4, -1, 8, -1, 4, 4, 0)/90.$$

Fichier C :/EF/z_info-quadratures/quadratures_T.m

```

clear
%Matrice et second membre du systeme :
A = sym('A');
f = sym('f');

%Ordre n de la formule de quadrature :
n = 4;

%Calcul des coordonnees barycentriques L1, L2, L3 dans l'ordre
% des points de la formule de quadrature :
%Calcul et de L1 et affichage de n*L1 :
c = 1;
for k = 0 :n
    for i = 0 :n - k
        L1(1,c) = (n - i - k)/n;
        c = c + 1;
    end;
end;
disp(['Valeurs de n*L1 : ', num2str(n*L1)]);

%Calcul et de L2 et affichage de n*L2 :
c = 1;
for k = 0 :n
    for i = 0 : n - k
        L2(1,c) = i/n;
        c = c + 1;
    end;
end;
disp(['Valeurs de n*L2 : ', num2str(n*L2)]);

%Calcul et de L3 et affichage de n*L3 :
c = 1;
for k = 0 :n
    for i = 0 :n - k
        L3(1,c) = k/n;
        c = c + 1;
    end;
end;

```

```

end ;
disp(['Valeurs de n*L3 : ', num2str(n*L3)]) ;

%Calcul des termes du systeme :
for j=1 :(n + 1)*(n + 2)/2
    for p=1 :(n + 1)*(n + 2)/2
        A(j,p) = ((L1(p)))^(n*(L1(j)))*((L2(p)))^(n*(L3(j))) ;
    end ;
    f(j,1) = factorial(n*(L1(j)))*factorial(n*(L3(j))) ...
            /factorial((2 + n*(L1(j))+ n*(L3(j)))) ;
end ;

%Resolution du systeme :
C = A\f ;

%Valeur des coefficients de la formule de quadrature :
disp('Valeurs des coefficients de la formule de quadrature : ')
disp(C') ;

```

Les calculs sont exacts avec ce fichier jusqu'à $n = 8$. Des erreurs apparaissent pour $n = 10$. Pour cette valeur on utilisera le fichier `.mws`.

L'algorithme de la fonction s'écrit finalement, en utilisant les coefficients des formules de quadrature calculés, par exemple, à l'aide du programme de calcul formel ci-dessus (les formules de quadrature utilisées sont classiques et on trouvera la valeur de ces coefficients dans la littérature sur le sujet) :

- Initialisation des coordonnées des sommets du triangle K de sommets $a = (a_1, a_2)$, $b = (b_1, b_2)$, $c = (c_1, c_2)$:

```
a1 = X(1) ;
```

```
b1 = X(2) ;
```

```
c1 = X(3) ;
```

```
a2 = Y(1) ;
```

```
b2 = Y(2) ;
```

```
c2 = Y(3) ;
```

- Calcul du déterminant de la matrice Jacobienne :

```
de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;
```

Choix de la formule de quadrature suivant la valeur de `D.intT`

Si `D.intT = 4`

- $$\begin{aligned} & (4a + 0b + 0c, 3a + 1b + 0c, 2a + 2b + 0c, 1a + 3b + 0c, 0a + 4b + 0c)/4, \\ & (3a + 0b + 1c, 2a + 1b + 1c, 1a + 2b + 1c, 0a + 3b + 1c)/4, \\ & (2a + 0b + 2c, 1a + 1b + 2c, 0a + 2b + 2c)/4, \\ & (1a + 0b + 3c, 0a + 1b + 3c)/4, \\ & (0a + 0b + 4c)/4. \end{aligned}$$

$$\begin{aligned} L1 &= [4, 3, 2, 1, 0, 3, 2, 1, 0, 2, 1, 0, 1, 0, 0]/4; \\ L2 &= [0, 1, 2, 3, 4, 0, 1, 2, 3, 0, 1, 2, 0, 1, 0]/4; \\ L3 &= [0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 3, 3, 4]/4; \end{aligned}$$

```
coeff = abs(de)*...
      [0, 4, -1, 4, 0, 4, 8, 8, 4, -1 ,8, -1, 4, 4, 0]/90 ;
```

```
L1 = [5, 4, 3, 2, 1, 0, 4, 3, 2, 1, 0, 3, 2, 1, 0, 2, 1, 0, ...
      1, 0, 0]/5 ;
L2 = [0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 0, 1, 2, 3, 0, 1, 2, ...
      0, 1, 0]/5 ;
L3 = [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, ...
      4, 4, 5]/5 ;
coeff = abs(de)*[11, 25, 25, 25, 25, 11, 25, ...
                 200, 25, 200, 25, 25, 25, 25, ...
                 25, 25, 200, 25, 25, 25, 11]/2016 ;
```

1109

- Calcul des coordonnées cartésiennes des points de la formule de quadrature. On applique la formule

$$m = \ell_1(m)a + \ell_2(m)b + \ell_3(m)c,$$

où ℓ_1 , ℓ_2 et ℓ_3 sont les coordonnées barycentriques relatives aux sommets a , b et c ce qui donne

$$A_c = a_1 * L_1 + b_1 * L_2 + c_1 * L_3 ;$$

$$B_c = a_2 * L_1 + b_2 * L_2 + c_2 * L_3 ;$$

Fonction :

```
function [a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3]...
    = assemble_coeff_A(X, Y, c, D) ;
```

renvoie les coefficients et les coordonnées des points de la formule de quadrature numéro D.intA, pour l'intégration sur les arêtes du triangle K de sommets donnés par les vecteurs lignes ou colonnes X et Y . Les formules de quadrature actuellement programmées ont pour numéro 4, 6 ou 10 qui est aussi par construction leur ordre respectif *a priori* sur chaque arête : en fait ces formules sont respectivement d'ordre 5, 7 et 11 (cas des formules avec un nombre de points de quadrature impair).

Soient A_1 l'arête numéro 1 de sommets $(X(1), Y(1))$, $(X(2), Y(2))$, A_2 l'arête numéro 2 de sommets $(X(1), Y(1))$, $(X(3), Y(3))$ et A_3 l'arête numéro 3 de sommets $(X(2), Y(2))$, $(X(3), Y(3))$, soit c le vecteur de $\mathbb{R}^3$ de valeur donné par c . Pour une fonction f représentée, par exemple, par une variable f de type `inline`, alors l'intégrale

$$c(1) \int_{A_1} f ds + c(2) \int_{A_2} f ds + c(3) \int_{A_3} f ds$$

est approchée par `sum(coeff.*f(Ac, Bc))`.

Une formule d'intégration numéro 106 qui est une formule d'intégration dont les points de la formule de quadrature sont situés à l'intérieur, strictement, des arêtes a aussi été ajoutée. Cette formule est utilisée au Paragraphe 2.7.2 (*cf.* aussi l'utilisation de cette formule aux Figures 2.7.2-5, 2.7.2-6 et 2.7.2-7), pour l'étude d'un problème de transport. Pour une intégration sur l'intervalle $[0, 1]$, les points d'intégration sont obtenus par subdivision de l'intervalle $[0, 1]$ en intervalles de longueur $1/6$ (d'où le choix du numéro 106) : $(1/6, 2/6, 3/6, 4/6, 5/6)$, les coefficients associés $(11/20, -7/10, 13/10, -7/10, 11/20)$, avec un ordre de 5 pour cette formule.

Variables d'entrée :

Ce sont les mêmes que les variables de la fonction `assemble_coeff_T` auxquelles on ajoute la variable de pondération sur les arêtes c . Pour la variable D , c'est le champ $D.intA$ qui indique cette fois le numéro de la formule d'intégration.

X, Y : `DOUBLE ARRAY`. Vecteurs lignes ou colonnes telles que $(X(1), Y(1))$, $(X(2), Y(2))$, $(X(3), Y(3))$ sont les coordonnées des trois sommets du triangle courant. On note K ce triangle sur lequel on calcule les formules de quadrature.

c : `DOUBLE ARRAY`. Vecteur ligne de trois termes indiquant la pondération de chaque arête sur laquelle on intègre. Ce vecteur permet ainsi de choisir sur quelles arêtes on intègre. Ainsi pour calculer l'intégrale de f sur la première arête, on pose $c = (1, 0, 0)$, sur la seconde on pose $c = (0, 1, 0)$ et sur la troisième $(0, 0, 1)$.

D : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ D du Tableau 2.2.1-1c, Paragraphe

2.2.1. Le champ `D.intA` donne le numéro de la formule d'intégration. Les autres champs de `D` sont facultatifs et non utilisés. Dans la version actuelle de la fonction `D.intA` doit prendre une valeur de l'ensemble $\{4, 6, 10, 106\}$.

Variables de sortie :

Ce sont les mêmes variables que dans la fonction `assemble_coeff_T`, mais relativement à l'intégration sur les arêtes.

`a1, b1, c1, a2, b2, c2` : `DOUBLE ARRAY`. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c d'un triangle K .

Ici ces variables sont les coordonnées du triangle K courant : on a la relation $X = [a1, b1, c1]$ et $Y = [a2, b2, c2]$.

`de` : `DOUBLE ARRAY`. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c). Pour l'intégration sur les arêtes, cette variable n'intervient pas dans le calcul des coefficients de la formule quadrature.

`Ac, Bc` : `DOUBLE ARRAY`. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature.

`coeff` : `DOUBLE ARRAY`. Vecteur ligne des coefficients de la formule de quadrature.

`L1, L2, L3` : `DOUBLE ARRAY`. Vecteurs ligne des coordonnées barycentriques des points de la formule de quadrature.

Les exemples ci-dessous sont tirés du `help` de la fonction. Ils sont analogues aux 2 exemples donnés pour la fonction `assemble_coeff_T`.

Exemple 1 : Dans cet exemple on applique la formule de quadrature d'ordre 4 sur un triangle de sommets $a = (0, 0)$, $b = (1/2, 0)$, $c = (0, 1)$. On calcule la longueur de la première arête $\mathcal{A}_1$ de sommets a et b de ce triangle, puis toujours sur cette arête l'intégrale

$$\int_{\mathcal{A}_1} x(s) ds$$

qui vaut, en posant comme paramétrage de l'arête $\gamma(t) = (t, 0)$

$$\int_0^{1/2} \gamma_1(t) |\gamma'(t)| dt = \int_0^{1/2} x dx = 1/8,$$

et on calcule enfin le périmètre du triangle qui vaut

$$1/2 + 1 + \sqrt{5}/2 \simeq 2.6180.$$

Ensuite, on trace sur une figure les points de la formule de quadrature ainsi que leur ordre de numérotation. Le premier exemple du help de la fonction `assemble_coeff_A` est le suivant.

EXEMPLE 1

```
%Integration de deux fonctions et traces des points de la
%formule de la formule de quadrature utilisee pour integrer :
clear
close all
f = inline('x.^0','x','y');
g = inline('x','x','y');
X = [0,1/2,0];
Y = [0,0,1];
D.intA = 4;
[A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, L1, L2, L3] = ...
    assemble_coeff_A(X, Y, [1,0,0], D);
disp('Longueur de la premiere arete du triangle');
If = sum(COEFF.*f(Ac, Bc))
disp('Integrale de x sur la premiere arete : int_0^1/2 x dx');
Ig = sum(COEFF.*g(Ac, Bc))
[A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, L1, L2, L3] = ...
    assemble_coeff_A(X, Y, [1,1,1], D);
disp('Perimetre du triangle');
Ip = sum(COEFF.*f(Ac, Bc))
hold on
patch(X,Y,'w');
for i=1 :3,
    text(X(i) - 0.09,Y(i) - 0.2 + 0.35*(i==3), ...
        ['Sommet n ', num2str(i)], 'FontSize',16);
end;
for i=1 :length(Ac),
    text(Ac(i) + 0.02 - (i>5).*(i<11)*0.1, ...
```

```

    Bc(i) + 0.02 - (i<6)*0.1, num2str(i),'FontSize',16);
end;
plot(Ac,Bc,'o')
title(['Points de la formule de quadrature dans leur' ...
      ' ordre de numerotation']);
axis([-0.1,0.7,-0.4,1.2]);
%Eventuelle sauvegarde de la figure obtenue : enlever le %
%print('-dpng','C :/EF_text/figures/points_quad_aretes.png');

```

Les points de la formule de quadrature sont donnés à la figure 4.3-9.

Points de la formule de quadrature dans leur ordre de numérotation

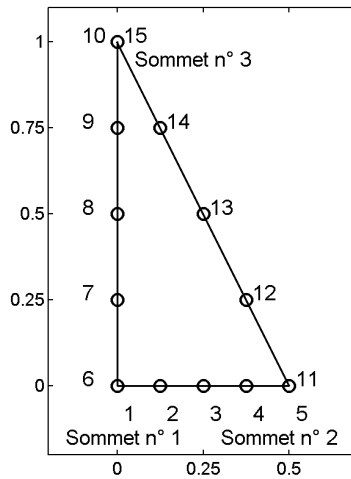


Figure 4.3-9.

Exemple 2 : On teste l'ordre des méthodes dans cet exemple. Soit K le triangle de sommets $(0,0)$, $(1/2,0)$, $(0,1)$. Sur l'arête δ de sommets $(0,0)$, $(1/2,0)$ on a avec le paramétrage $\gamma(t) = (t, 0)$ de l'arête :

$$\int_{\delta} x^i ds = \int_0^{1/2} \gamma_1(s)^i |\gamma'_1(s)| ds = \int_0^{1/2} x^i = \frac{1}{2^{i+1}(i+1)}.$$

Une formule de quadrature d'ordre n doit redonner cette valeur pour tout $1 \leq i \leq n$. Le second exemple du `help` de la fonction `assemble_coeff_A` est le suivant.

EXEMPLE 2

```

%Test de l'ordre des formules d'integration : L doit etre nul a
%la precision machine pres.
%On teste trois formules d'ordre resp. 4, 6 et 10
clear

```

```

for m = [4,6,10] ;
    L = [] ;
    D.intA = m ;
    %En fait ces formules sont d'ordre 5, 7, 11 (parite)
    n = D.intA + 1 ;
    X = [0,1/2,0] ;
    Y = [0,0,1] ;
    [A1, B1, C1, A2, B2, C2, DE, Ac, Bc, COEFF, ...
     L1, L2, L3] = assemble_coeff_A(X, Y, [1,0,0], D) ;
    for i=0 :n
        f = inline(['x.^',num2str(i)], 'x', 'y') ;
        L(i+1) = sum(COEFF.*f(Ac, Bc)) - (1/2)^(i + 1)/(1 + i) ;
    end ;
    disp(max(max(abs(L)))) ;
end ;

```

Algorithme :

EF/Assemblage_elementaire/assemble_coeff_A.m

On remet ici la partie déclarative de la fonction :

```

function [a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3]...
    = assemble_coeff_A(X, Y, c, D) ;

```

L'algorithme suit le même principe que celui utilisé pour l'intégration sur les triangles. Les coefficients des formules de quadrature sont calculés par le fichier

EF/z_info_quadratures/quadratures_A.mws.

comme indiqué au Paragraphe 3.3.2 où sont décrits le principe de la construction des formules de quadrature sur les arêtes utilisées dans `assemblage_coeff_A`, à titre d'exemple, le calcul détaillé des coefficients dans le cas de la formule de quadrature d'ordre 4 sur chaque arête, ainsi que d'autres variantes de fichiers `.mws` ou `.m` de calcul des coefficients.

Nous décrivons ci-dessous l'algorithme utilisé dans le fichier `quadratures_A.mws` et ses variantes. Soient a_1, a_2, a_3 les sommets d'un triangle K . On note A_1 l'arête numéro 1 de sommets a_1, a_2 avec le paramétrage $\gamma_1(t) = a_1 + t(a_2 - a_1), t \in [0, 1]$, A_2 l'arête numéro 2 de sommets a_1, a_3 de paramétrage $\gamma_2(t) = a_1 + t(a_3 - a_1), t \in [0, 1]$, et A_3 l'arête numéro 3 de sommets a_2, a_3 de paramétrage $\gamma_3(t) = a_2 + t(a_3 - a_2), t \in [0, 1]$. On note ℓ_1, ℓ_2 et ℓ_3 les coordonnées barycentriques relatives

à ces trois sommets, $\|\cdot\|_2$ la norme euclidienne et $d\ell$ l'élément de longueur. Pour construire une formule de quadrature d'ordre n on choisit les $3n+3$ points suivants :

$$m_p = \frac{n-p}{n}a_1 + \frac{p}{n}a_2, \quad m_{n+1+p} = \frac{n-p}{n}a_1 + \frac{p}{n}a_3, \\ m_{2n+2+p} = \frac{n-p}{n}a_2 + \frac{p}{n}a_3, \quad 0 \leq p \leq n.$$

On voit que l'ordre suivi pour le placement des points de la formule de quadrature, choix des arêtes successives et sens de parcours sur les arêtes, est le même que celui choisi pour les ddl d'arêtes sur les triangles. Pour $n = 4$, on peut visualiser l'ordre des points d'intégration sur la Figure 4.3-9 de l'exemple 1 ci-dessus.

En sortie de la fonction `assemblage_coeff_A`, $Ac(p+1)$ et $Bc(p+1)$ sont les coordonnées en x et y du point m_p et $L1(p+1)$, $L2(p+1)$ et $L3(p+1)$ ses coordonnées barycentriques données respectivement par $\ell_1(m_p)$, $\ell_2(m_p)$ et $\ell_3(m_p)$. Le vecteur `coeff(p)` est alors le coefficient de la formule de quadrature associé au point m_p . La valeur de ces coefficients est donnée directement dans l'algorithme de la fonction `assemble_coeff_A`.

Les coefficients de la formule de quadrature sont calculés de la manière suivante. On note $(c_p)_{0 \leq p \leq 3n+2}$ le vecteur des coefficients associé m_p dans la formule de quadrature *i.e.* :

$$I(f) = c(1) \sum_{p=0}^n c_p f(m_p) + c(2) \sum_{p=n+1}^{2n+1} c_p f(m_p) + c(3) \sum_{p=2n+2}^{3n+2} c_p f(m_p)$$

pour l'approximation de

$$I(f) = c(1) \int_{A_1} f d\ell + c(2) \int_{A_2} f d\ell + c(3) \int_{A_3} f d\ell \\ = c(1) \|a_2 - a_1\|_2 \int_0^1 f(\gamma_1(t)) dt + c(2) \|a_3 - a_1\|_2 \int_0^1 f(\gamma_2(t)) dt \\ + c(3) \|a_3 - a_2\|_2 \int_0^1 f(\gamma_3(t)) dt.$$

Pour que la méthode soit d'ordre n , il suffit que les coefficients c_p vérifient pour tout élément $\{\ell_1^r \ell_2^s\}_{0 \leq r+s \leq n}$ base de $P_n(K)$:

$$\sum_{p=0}^n c_p \ell_1(m_p)^r \ell_2(m_p)^s = \int_{A_1} \ell_1^r \ell_2^s d\ell = \|a_2 - a_1\| \int_0^1 \ell_1^r(\gamma_1(t)) \ell_2^s(\gamma_1(t)) dt$$

ce qui donne

$$\sum_{p=0}^n c_p \left(\frac{n-p}{n}\right)^r \left(\frac{p}{n}\right)^s = \int_{A_1} \ell_1^r \ell_2^s d\ell = \|a_2 - a_1\| \int_0^1 (1-t)^r t^s dt = \frac{\|a_2 - a_1\| r! s!}{(r+s+1)!}.$$

Comme $(1-t)^r t^s$ est un polynôme de degré inférieur ou égal à n , il faut et il suffit, par linéarité de l'intégrale et des formules de quadrature, que la relation ci-dessus soit vérifiée sur $\{(1-t)^r t^s\}_{r+s=n}$ qui est une base de l'espace vectoriel des polynômes de degré inférieur ou égal à n soit :

$$\sum_{p=0}^n c_p \left(\frac{n-p}{n}\right)^r \left(\frac{p}{n}\right)^s = \frac{\|a_2 - a_1\| j!(n-j)!}{(n+1)!}, \quad 0 \leq j \leq n.$$

Pour les deux autres arêtes il suffit de même d'avoir :

$$c_{2n+2+p} = c_{n+1+p} = c_p, \quad 0 \leq p \leq n,$$

qui est le choix utilisé dans `assemble_coeff_A`. Posons

$$x_p = (n-p)/n, \quad 0 \leq p \leq n$$

et $(\hat{c}_p)_{0 \leq p \leq n}$ le vecteur de composantes :

$$\hat{c}_p = \|a_2 - a_1\|^{-1} c_p, \quad 0 \leq p \leq n.$$

Le vecteur $\hat{c}$ est alors solution du système

$$\sum_{p=0}^n \hat{c}_p x_p^j (1-x_p)^{n-j} = \frac{j!(n-j)!}{(n+1)!}, \quad 0 \leq j \leq n.$$

système qui s'écrit :

$$A\hat{c} = f,$$

avec

$$A_{jp} = x_p^j (1-x_p)^{n-j}, \quad 0 \leq j, p \leq n \quad \text{et} \quad f_j = \frac{j!(n-j)!}{(n+1)!}, \quad 0 \leq j \leq n.$$

C'est ce système qui est calculé et résolu dans le fichier :

`EF/z_info_quadratures/quadratures_A.mws`

ou avec le fichier

`EF/z_info_quadratures/int_D1.m`.

- Soient r et s définies par $r(p) = n\ell_1(m_p)$ et $s(p) = n\ell_3(m_p)$, le système ci-dessus s'écrit aussi de manière similaire au cas des formules de quadrature sur les triangles

$$\sum_{p=0}^n \hat{c}_p \ell_1(m_p)^{r(j)} \ell_2(m_p)^{s(j)} = \int_0^1 \ell_1^{r(j)} \ell_2^{s(j)} = \frac{r(j)!s(j)!}{(1+r(j)+s(j))!}, \quad 0 \leq j \leq n,$$

r prenant ici les valeurs $r(j) = n - j$ et $s(j) = 0$, $0 \leq j \leq n$.

C'est ce système qui est calculé pour les formules d'ordre 4, 6 et 10 dans les fichiers respectifs :

```
C : /EF/z_info_quadratures/quadrature_A4.mws,
C : /EF/z_info_quadratures/quadrature_A6.mws,
C : /EF/z_info_quadratures/quadrature_A10.mws.
```

- Ci-dessous nous avons mis l'équivalent du fichier **quadratures_A.mws** en version *.m*. Dans ce fichier on calcule, pour $n = 4$, les valeurs respectives des coordonnées barycentriques $\ell_1(m_p)$ et $\ell_2(m_p)$, $0 \leq p \leq 4$, de la formule de quadrature :

$$\mathbf{x} = (1, 3/4, 1/2, 1/4, 0), \mathbf{y} = (0, 1/4, 1/2, 3/4, 1)$$

puis la matrice **A** et le second membre **f** du système dont la résolution donne la valeur des coefficients de la formule de quadrature :

$$\mathbf{C} = (7/90, 16/45, 2/15, 16/45, 7/90).$$

Fichier C :/EF/z_info_quadratures/quadratures_A.m

```
clear ;
syms('A') ;
syms('f') ;
n = 4 ;

%Subdivision de [0,1] en n intervalles :
for p=0 :n,
    x(p+1) = (n - p)/n ;
end ;
for p=0 :n
    y(p+1) = p/n ;
end ;

%Initialisation de la matrice et du second membre
% du systeme :
for j=0 :n
    for p=0 :n
        A(j+1,p+1) = x(p+1)^j*y(p+1)^(n - j) ;
    end ;
```



```

    f(j+1,1) = factorial(j)*factorial(n - j)/...
                factorial(n + 1);
end;
%Calcul des coefficients de la formule de quadrature :
C = A\f;

%Affichage des resultats :
disp('Points de la formule de quadrature sur [0,1] : ')
disp(rats(fliplr(x)));

disp(['Valeur des coefficients associes de', ...
      ' la formule de quadrature :']);
disp(fliplr(C'));

%Verification des formules sur l'integrale de x^i :
X = [0 :n]/n;
C = numeric(C');
disp(['Verification : calcul de l''ordre ', ...
      ' maximal (', num2str(n + 1),')']);
for i=0 :n+1
    disp(['degre ', num2str(i), ' : ', ...
          num2str(sum(C.*X.^i) - 1/(i+1))]);
end;

```

- Nous avons aussi ajouté une version qui permet de calculer des formules de quadrature avec des points intérieurs comme la formule numéro 106. Dans ce fichier on calcule pour $n = 6$, les valeurs respectives des coordonnées barycentriques $\ell_1(m_p)$ et $\ell_2(m_p)$, $1 \leq p \leq 5$, de la formule de quadrature :

$$x = (5/6, 2/3, 1/2, 1/3, 1/6), y = (1/6, 1/3, 1/2, 2/3, 5/6)$$

et on résout le système

$$A\hat{c} = f,$$

avec, pour une formule *a priori* d'ordre $n - 2$:

$$A_{jp} = x_p^j (1 - x_p)^{n-2-j}, \quad 0 \leq j, p \leq n - 2 \quad \text{et} \quad f_j = \frac{j!(n-2-j)!}{(n-1)!}, \quad 0 \leq j \leq n - 2.$$

pour cela on calcule la matrice **A** et le second membre **f** du système dont la résolution donne la valeur des coefficients de la formule de quadrature :

$$C = (11/20, -7/10, 13/10, -7/10, 11/20).$$

Le fichier pour le calcul de formules avec points intérieurs est le suivant.

Fichier C : /EF/z_info-quadratures/quadratures_A_int.m

```

clear ;
syms('A') ;
syms('f') ;
n = 6 ;
%Subdivision de [1/n,(n - 1)/n] en n - 2 intervalles :
for p=1 :n-1
    x(p) = (n - p)/n ;
end ;
for p=1 :n-1
    y(p) = p/n ;
end ;

%Initialisation de la matrice et du second membre
% du systeme :
for j=0 :n-2
    for p=1 :n-1
        A(j+1,p) = x(p)^j*y(p)^(n - 2 - j) ;
    end ;
    f(j+1,1) = factorial(j)*factorial(n - 2 - j)/ ...
        factorial(n - 1) ;
end ;

%Calcul des coefficients de la formule de quadrature :
C = A\f ;

%Affichage des resultats :
disp('Points de la formule de quadrature sur [0,1] : ')
disp(rats(fliplr(x))) ;

disp(['Valeur des coefficients associes de', ...
      ' la formule de quadrature :']) ;
disp(fliplr(C')) ;

%Verification des formules sur l'integrale de x^i :
X = [1 :n-1]/n ;
C = numeric(C') ;
disp(['Verification : calcul de l''ordre ', ...

```

```

        'maximal (', num2str(n - 1),')')]);
for i=0 :n
    disp(['degre ', num2str(i), ' : ', ...
        num2str(sum(C.*X.^i) - 1/(i+1))]);
end ;

```

Les résultats montrent que l'on obtient une formule d'ordre $n - 1 = 5$

Dans la méthode ci-dessus, les calculs sont effectués sur les fonctions du $P_n(K)$, comme pour l'intégration sur le triangle. Plus simplement, on peut aussi partir de la base $\{(1 - t)^{n-j}\}_{0 \leq j \leq n}$ comme ci-dessous (ou bien $\{(t)^j\}_{0 \leq j \leq n}$) de l'espace vectoriel des polynôme de degré inférieur ou égal à n . Ce qui donne $\hat{c}$ solution de

$$A\hat{c} = f$$

avec

$$A_{jp} = \ell_1(m_p)^{n-j} = x_p^{n-j} = \binom{n-p}{n}^{n-j}, \quad f(j) = \frac{1}{n-j+1}, \quad 0 \leq j \leq n, \quad 0 \leq p \leq n.$$

On met ci-dessous un fichier *.m* correspondant à cet l'algorithme :

Fichier C :/EF/z_info_quadratures/quadratures_Ab.m

```

clear ;
syms('A') ;
syms('f') ;
n = 4 ;

%Subdivision de [0,1] en n intervalles :
for p=0 :n,
    x(p+1) = (n - p)/n ;
end ;

%Initialisation de la matrice et du second membre
% du systeme :
for j=0 :n
    for p=0 :n
        A(j+1,p+1) = x(p+1)^(n - j) ;
    end ;
    f(j+1,1) = 1/(n - j + 1) ;
end ;

```

```

%Calcul des coefficients de la formule de quadrature :
C = A\f ;

%Affichage des resultats :
disp('Points de la formule de quadrature sur [0,1] : ')
disp(rats(fliplr(x)));

disp(['Valeur des coefficients associes de', ...
      ' la formule de quadrature :']);
disp(fliplr(C'));

%Verification des formules sur l'integrale de x^i :
X = [0 :n]/n;
C = numeric(C');
disp(['Verification : calcul de l''ordre ', ...
      'maximal (', num2str(n + 1),')']);
for i=0 :n+1
    disp(['degre ', num2str(i), ' : ', ...
          num2str(sum(C.*X.^i) - 1/(i+1))]);
end ;

```

L'algorithme de la fonction `assemble_coeff_A` s'écrit finalement, les coefficients des formules de quadrature étant déjà calculés, de la manière suivante :

- Sommets du triangle K de sommets $a = (a_1, a_2)$, $b = (b_1, b_2)$, $c = (c_1, 2)$:

```

a1 = X(1) ;
b1 = X(2) ;
c1 = X(3) ;
a2 = Y(1) ;
b2 = Y(2) ;
c2 = Y(3) ;

```

- Déterminant de la matrice Jacobienne :

```

de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]' ;

```

- Longueur des arêtes, représente aussi la mesure ℓ :

```

DGab = sqrt((b1 - a1).^2 + (b2 - a2).^2) ;
DGac = sqrt((c1 - a1).^2 + (c2 - a2).^2) ;
DGbc = sqrt((c1 - b1).^2 + (c2 - b2).^2) ;

```

Suivant la valeur de `D.intT`

Si D.intA = 4

- Calcul des valeurs relatives à la formule d'intégration numéro 4 qui est aussi d'ordre 4 (et même d'ordre 5).

- Les points d'intégration vu-ci-dessus sont dans l'ordre suivant

$$(4a + 0b + 0c, 3a + 1b + 0c, 2a + 2b + 0c, 1a + 3b + 0c, 0a + 4b + 0c)/4,$$

$$(4a + 0b + 0c, 3a + 0b + 1c, 2a + 0b + 2c, 1a + 0b + 3c, 0a + 0b + 4c)/4,$$

$$(0a + 4b + 0c, 0a + 3b + 1c, 0a + 2b + 2c, 0a + 1b + 3c, 0a + 0b + 4c)/4.$$

La numérotation pour les autres formules est analogue. Les coordonnées barycentriques au points de la formule de quadrature valent donc :

$$L1 = [4, 3, 2, 1, 0, 4, 3, 2, 1, 0, 0, 0, 0, 0, 0]/4 ;$$

$$L2 = [0, 1, 2, 3, 4, 0, 0, 0, 0, 0, 4, 3, 2, 1, 0]/4 ;$$

$$L3 = [0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 0, 1, 2, 3]/4 ;$$

- Concaténation des 3 vecteurs successifs donnés pour chaque arête A_i par le vecteur des coefficients de la formule de quadrature pour une arête de longueur 1 multiplié par la longueur de l'arête A_i , $1 \leq i \leq 3$ et pondéré par $c(i)$:

$$\text{coeff} = [c(1)*DGab*[7, 32, 12, 32, 7]/90, \dots$$

$$c(2)*DGac*[7, 32, 12, 32, 7]/90, \dots$$

$$c(3)*DGbc*[7, 32, 12, 32, 7]/90] ;$$

Si D.intT = 6

- Sur le même principe, la formule numéro 6 est donnée par :

$$L1 = [6, 5, 4, 3, 2, 1, 0, 6, 5, 4, 3, 2, 1, 0, 0, 0, 0, \dots \\ 0, 0, 0, 0]/6 ;$$

$$L2 = [0, 1, 2, 3, 4, 5, 6, 0, 0, 0, 0, 0, 0, 0, 6, 5, 4, \dots \\ 3, 2, 1, 0]/6 ;$$

$$L3 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2, \dots \\ 3, 4, 5, 6]/6 ;$$

$$cv = [164, 864, 108, 1088, 108, 864, 164]/3360 ;$$

$$\text{coeff}=[c(1)*DGab*cv, c(2)*DGac*cv, c(3)*DGbc*cv] ;$$

Les cas D.intT = 10 et D.intT = 106 se traitent de manière analogue. On pourra consulter la fonction `assemble_coeff_A` pour les valeurs des coefficients.

- Calcul des coordonnées cartésiennes des points de la formule de quadrature. On applique la formule

$$m = \ell_1(m)a + \ell_2(m)b + \ell_3(m)c,$$

où ici ℓ_1 , ℓ_2 et ℓ_3 sont les coordonnées barycentriques relatives aux sommets a_1 , b et c , ce qui donne :

$$Ac = a1*L1 + b1*L2 + c1*L3 ;$$

$$Bc = a2*L1 + b2*L2 + c2*L3 ;$$

Fonction :

```
function [Sc_Dom] = matrice_elem_T(donnees, i, T, Tc, X, Y, ...
                                   Uc, D, Mat_syst_Dom, Ass_Dom, Sc_Dom) ;
```

Calcul de matrices élémentaires pour l'assemblage sur les triangles de la forme bilinéaire associée à l'EDP. Ce sont les matrices élémentaires correspondantes au terme en a_Ω de la forme bilinéaire $\mathcal{A}$ sur $E \times E$ dans (1.5) qui sont ici calculées.

Pour un triangle donné (désigné ci-dessus par son numéro i), cette fonction calcule la matrice élémentaire relative à la forme bilinéaire associée au système d'EDP (hors termes d'arêtes). La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP et aux triangles (désigné ci-dessus par Sc_Dom , matrice de type CELL).

Cette fonction est appelée par la fonction `EF_assemblage` dans la boucle de calcul de la matrice du système EF associée à l'EDP. Dans cette boucle est ajoutée successivement à la matrice Sc_Dom toutes les contributions de chaque triangle du maillage à cette matrice en dehors des contributions dues aux termes d'arête.

Dans la fonction `EF_assemblage` la contribution de Sc_Dom , calculée par `matrice_elem_T` (pour les triangles), de Sc_Fr , calculée par `matrice_elem_A` (pour les arêtes frontières), de Sc_Ad , calculée par `matrice_elem_Ad` (pour les arêtes) et par `matrice_elem_Ad_int_LR` (pour la méthode de Lesaint-Raviart), sont ajoutées entre-elles pour former la matrice totale d'assemblage Sc de la forme bilinéaire $\mathcal{A}$ sur $E \times E$, matrice qui est ensuite transformée pour tenir compte des conditions au bord et de l'élimination d'éventuelles d'inconnues comme par exemple dans le cas d'espaces quotients ou d'équations identiques dans le système.

La commande

```
[Sc_Dom] = matrice_elem_T(donnees, i, T, Tc, X, Y, Uc, D, ...
                          Mat_syst_Dom, Ass_Dom, Sc_Dom) ;
```

permet de calculer la matrice élémentaire, par blocs, du système relativement au triangle numéro i , que l'on notera aussi i , de sommets $T(i, 1 : 3)$ et l'ajoute à la matrice globale du système représentée par la variable Sc_Dom qui est une matrice de type CELL pour la représentation par blocs de la matrice du système. Le terme `donnees` est la variable de type `struct` qui représente les données du problème. La variable i est le numéro du triangle courant, T est le tableau qui décrit le maillage, Tc représente les degrés de liberté des différents types d'espaces, X et Y les coordonnées en x et en y des sommets du maillage, Uc est un tableau de cellules représentant un élément de l'espace EF produit E comme par exemple un itéré de la solution dans un point fixe, D est une variable de type `struct` décrivant les espaces d'EF et les formules d'intégration, Mat_syst_Dom un handle de fonction (ou

un nom de fonction) décrivant le système d'équations aux dérivées partielles sur le domaine, **Ass_Dom** les valeurs des fonctions de base ainsi que de leurs dérivées aux points de la formule de quadrature sur le triangle courant. En général cette variable est calculée par **EF_prepa** avec la fonction **matrice_elem_prepa_T**, ou de manière plus directe, avec la fonction **pre_assemble_T** comme dans les deux exemples qui suivent. La variable de sortie **Sc_Dom** est un tableau de type **CELL** qui représente la matrice globale d'assemblage.

Variables d'entrée :

donnees : **STRUCT ARRAY**. Variable décrivant les données du problème EF. Dans cette fonction cette variable est utilisée (uniquement) comme argument d'entrée de la fonction désignée par la variable **Mat_syst_Dom**, qui est une fonction définie par l'utilisateur.

i : **DOUBLE ARRAY**. Numéro du triangle courant.

T : **DOUBLE ARRAY**. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Tc : **CELL ARRAY**. Vecteur ligne de cellules. Les cellules sont des matrices de type **DOUBLE ARRAY**. La matrice **Tc**{*i*} décrit triangle par triangle les *ddl* du *i*-ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

X, Y : **DOUBLE ARRAY**. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Uc : **CELL ARRAY**. Vecteur colonne dont les cellules sont des vecteurs colonnes de type **DOUBLE ARRAY**. Représente les composantes dans sa base d'un élément de l'espace EF produit courant *E* (*cf.* Eq. (1.24) du Paragraphe 1.5). Ce terme figure, par exemple, la solution du problème EF ou un itéré courant dans une méthode itérative. Dans le cas des espaces à base nodale (espaces P_k^c ou P_k^d), valeur des composantes et valeur de la fonction aux *ddl* correspondants sont identiques.

D : **STRUCT ARRAY**. Variable de description des espaces d'EF correspondant à **donnees.D**. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe 2.2.1. En pratique **D = donnees.D**. Les champs de **D** utilisés sont les suivants :

D.E : Choix des types d'espaces.

D.der = [0,1,2] : Choix des dérivations. Toujours fixe pour ce code.

D.intT : Choix de la formule de quadrature dans l'appel de **assemble_coeff_T**.

D.f : Vecteur ligne. **D.f**(*i*) est un numéro permettant de choisir un espace d'EF parmi ceux de la liste donnée par le vecteur **D.E** : le *i*-ème espace de l'espace EF produit *E* a pour numéro **D.E**(**D.f**(*i*)).

Mat_syst_dom : **FUNCTION_HANDLE ARRAY**. Handle sur la fonction qui définit les EDP du premier membre sur Ω et qui correspond au terme a_Ω qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feq_Dom** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **Mat_syst_dom**

= `donnees.feq_Dom`. La syntaxe de la fonction désignée par `Mat_syst_Dom`, que nous notons ci-dessous `eq_Dom`, est la suivante :

Syntaxe de la fonction appelée par `matrice_elem_T`.

```
function [Mc, Nc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, c2, ...
                           de, Ac, Bc, numK) ;
```

avec :

Variables d'entrée (pour `eq_Dom`) :

`donnees` : STRUCT ARRAY. Variable décrivant les données du problème EF.

`uc` : CELL ARRAY. Tableau $n \times 3$ de cellules. Pour $i = 1, \dots, n$ ($n = \dim(E)$ est le nombre d'inconnues) et $j = 1, 2, 3$ (j indique la dérivation $j - 1$), `uc`{ i, j } est un vecteur ligne de type double array. Pour $j = 0, 1, 2$, le terme `uc`{ $i, j + 1$ } représente les valeurs, aux points de la formule de quadrature utilisée pour l'intégration sur le triangle courant, de la dérivée d'ordre $j = 0, 1, 2$ de la i -ème composante de l'itéré courant X^0 de (1.5) (et (1.10) pour le second membre), représenté ici par le vecteur `Uc`. Ces valeurs sont données par $X_{i,j}^0(\mathbf{Ac}, \mathbf{Bc})$. Cette variable est décrite au Tableau 2.2.4-1, première partie, du Paragraphe 2.2.4.

`a1, b1, c1, a2, b2, c2` : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c du triangle K de numéro `numK`.

`de` : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

`Ac, Bc` : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature. Il s'agit ici de la formule de quadrature de numéro donné par `D.intT`.

`numK` : DOUBLE ARRAY. Scalaire, désigne le numéro de triangle courant auquel appartiennent les points de coordonnées cartésiennes `Ac` et `Bc`.

Variable de sortie (pour `eq_Dom`) :

`Mc, Nc` : CELL ARRAY. Les variables de sortie `Mc` et `Nc` sont les variables de type `cell` décrivant l'EDP sur le domaine. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de `matrice_elem_T` :

Ass_Dom : CELL ARRAY. Tableau de cellules $N \times N_E \times 3$ où N est le nombre de triangles et $N_E = \text{length}(\text{donnees.D.E})$ est le nombre de type d'espace utilisés (cf. le début du Paragraphe 2.2). Le terme **Ass_Dom** $\{i, j, k\}$ est le tableau des valeurs, aux points de la formule de quadrature sur le triangle K numéro i , de la dérivée d'ordre $k-1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par **donnees.D.E**(j). Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.D.intT** utilisée sur les triangles et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ **Ass_Dom** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Sc_Dom : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Représentation de la matrice par blocs correspondant à l'assemblage de la partie en Ω de la forme bilinéaire $\mathcal{A}$ (terme en a_Ω) de (1.5), Paragraphe 1.2. La forme bilinéaire est définie par la fonction désignée par le champ **donnees.feq_Dom** (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage se fait par boucle sur les triangles, le rôle de la fonction **matrice_elem_T** est de calculer la contribution du triangle courant à la forme bilinéaire. Le terme **Sc_Dom** $\{i, j\}$ est la matrice de taille $\dim(E_i) \times \dim(E_j)$ qui correspond à la matrice d'assemblage de la restriction de la partie de la forme bilinéaire qui porte sur $E_j \times E_i$, E_k désignant le k -ième espace de l'espace EF produit E et de numéro d'espace **donnees.D.E**(**donnees.D.f**(k)).

Variable de sortie (de `matrice_elem_T`) :

Sc_Dom : CELL ARRAY. Matrice du système augmentée de la matrice élémentaire calculée par **matrice_elem_T**.

Exemple 1 : Soit Ω le carré unité $[0,1]^2$ maillé par le triangle K_1 de sommets $(0,1)$, $(0,0)$ et $(1,0)$ et le triangle K_2 de sommets $(0,1)$, $(1,0)$ et $(1,1)$ (cf. la figure Figure 4.3-10 ci-après). On définit le maillage et la frontière par les valeurs suivantes des variables T, X, Y et Front :

```
%Tableau des sommets triangles par triangles
```

```
T = [1,2,3;1,3,4] ;
```

```
%Coordonnees des DDL
```

```
X = [0,0,1,1] ;
```

```
Y = [1,0,0,1] ;
```

```
%Frontiere
```

```
Front{1} = [1,2,3,4,1] ;
```

Le maillage résultant ainsi que la numérotation globale des sommets du maillage et la numérotation locale des sommets des triangles sont représentés à la figure 4.3-10 suivante.

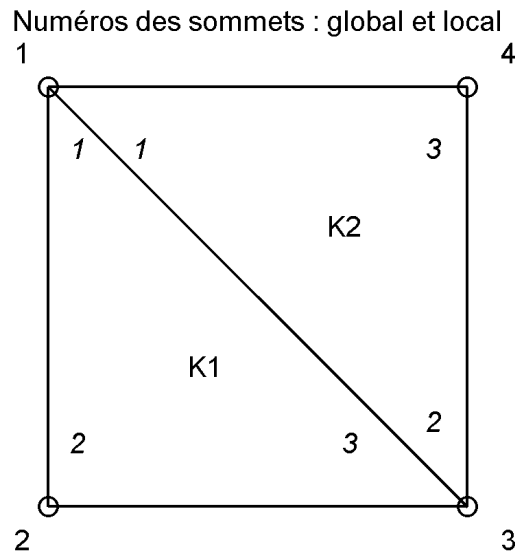


Figure 4.3-10. Numérotation des sommets du maillages (chiffres de 1 à 4 en caractères droits) et numérotation locale des triangles (chiffres en italique).

On pose E l'espace EF produit donné par $E = P_1^c \times P_2^c \times P_1^c$ (on utilisera aussi la notation $P_1^c - P_2^c - P_1^c$) sur ce maillage. Les numéros des *ddl* des espaces d'EF P_1^c et P_2^c que donne ce maillage sont indiqués à la figure 4.3-11 suivante. Les *ddl* globaux de E sont, dans l'ordre, les 4 *ddl* de l'espace d'EF P_1^c (qui sont les sommets de la Figure 4.3-10), puis les 9 *ddl* de l'espace d'EF P_2^c (formés des *ddl* de l'espace d'EF P_1^c et des points milieux d'arêtes, en suivant l'ordre de numérotation des arêtes du maillage) et enfin les 4 *ddl* de l'espace d'EF P_1^c .

Nous notons $\{d_i\}_{1 \leq i \leq 17}$ les *ddl* de E , $\{d_i\}_{1 \leq i \leq 4}$ sont les 4 *ddl* du P_1^c , $\{d_i\}_{5 \leq i \leq 13}$ sont, dans leur ordre, les 9 *ddl* du P_2^c et enfin $\{d_i\}_{14 \leq i \leq 17}$ sont donnés à nouveau par les 4 *ddl* successifs du P_1^c .

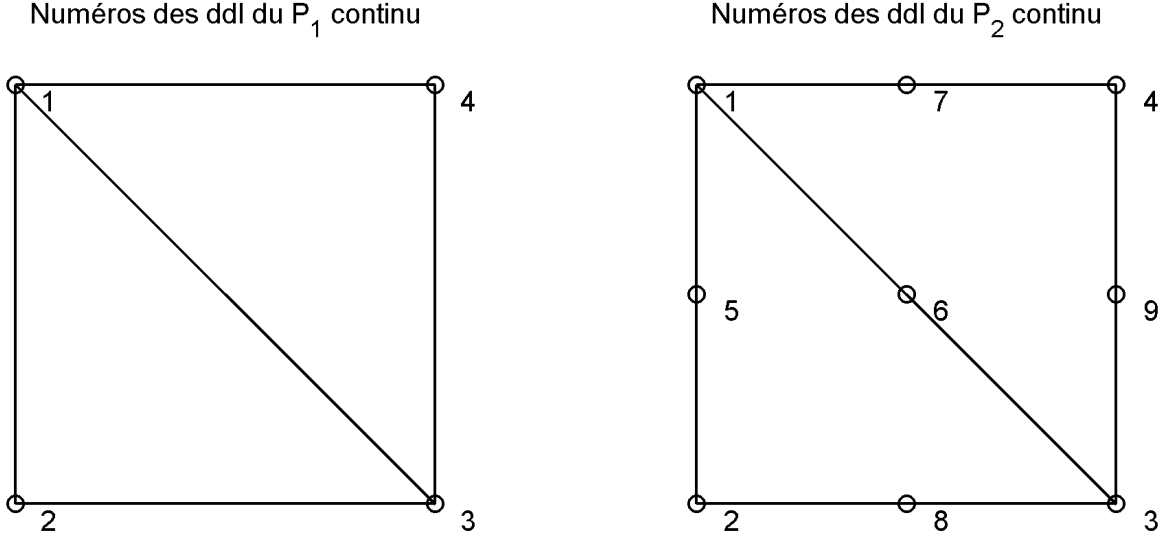


Figure 4.3-11. Degrés de liberté du P_1 continu et du P_2 continu.

Soit $f = (f_1, f_2, f_3)$ une fonction de Ω à valeur dans $\mathbb{R}^3$, on considère le problème $\exists u = (u_1, u_2, u_3)$ tel que

$$u_1 = f_1, u_2 = f_2, u_3 = f_3 \text{ sur } \Omega,$$

que l'on approche par le problème EF suivant :

$\exists u = (u_1, u_2, u_3) \in E$ tel que

$$\int_{\Omega} u_1 v_1 = \int_{\Omega} f_1 v_1, \int_{\Omega} u_2 v_2 = \int_{\Omega} f_2 v_2, \int_{\Omega} u_3 v_3 = \int_{\Omega} f_3 v_3, \forall v = (v_1, v_2, v_3) \in E.$$

Ce problème se met sous la forme équivalente suivante :

$$\exists u = (u_1, u_2, u_3) \in E \text{ tel que } a(u, v) = b(v), \forall v = (v_1, v_2, v_3) \in E$$

avec a la forme bilinéaire définie sur $E \times E$ par

$$a(u, v) = \int_{\Omega} u_1 v_1 + \int_{\Omega} u_2 v_2 + \int_{\Omega} u_3 v_3, \forall (u, v) \in E \times E$$

et b la forme linéaire définie sur E par

$$b(v) = \int_{\Omega} f_1 v_1 + \int_{\Omega} f_2 v_2 + \int_{\Omega} f_3 v_3, \forall v \in E.$$

Dans l'exemple qui suit, on assemble la matrice du système EF, notée A liée à la forme bilinéaire a . Le fichier exemple est la fonction *exemple1_matrice_elem_T.m* du répertoire EF/exemples/Assemblage_elementaire qui est mise en œuvre par la commande :

```
[Ass_Dom, Sc_Dom, A] = exemple1_matrice_elem_T ;
```

La matrice A du système est donnée en sortie par la variable A . Notons $\{\Phi_k\}_{1 \leq k \leq 15}$ les fonctions de la base nodale de E liés aux *ddl* de E . On calcule la matrice A en utilisant les formules suivante sur un triangle K de sommets a_i , $i = 1, 2, 3$:

$$\int_K \ell_{1,K}^i \ell_{2,K}^j = \int_K \ell_{1,K}^i \ell_{3,K}^j = \int_K \ell_{2,K}^i \ell_{3,K}^j = 2|K| \frac{i!j!}{(2+i+j)!},$$

où $\{\ell_{i,K}\}_{1 \leq i \leq 3}$ sont les coordonnées barycentriques relatives aux trois sommets $\{a_i\}_{1 \leq i \leq 3}$. Posons $\{\ell_{i,K_1}\}_{1 \leq i \leq 3}$ (resp. $\{\ell_{i,K_2}\}_{1 \leq i \leq 3}$) les coordonnées barycentriques relatives aux trois sommets numérotés 1, 2 et 3 du triangle K_1 (resp K_2), dans la numérotation locale des sommets sur K_1 (resp K_2) (voir la Figure 4.3-10 ci-dessus, les numéros étant en italique).

La fonction Φ_1 est la fonction de l'espace P_1^c qui vaut 1 au *ddl* numéro 1 du P_1^c et 0 aux autres *ddl* du P_1^c . On a alors

$$A(1, 1) = \int_{K_1} \Phi_1^2 + \int_{K_2} \Phi_1^2 = \int_{K_1} \ell_{1,K_1}^2 + \int_{K_2} \ell_{1,K_2}^2 = 1/6.$$

On trouve ainsi, en effectuant des calculs similaires sur les fonctions de base de l'espace P_1^c , les valeurs de A suivantes :

$$A(1 : 4, 1 : 4) = A(14 : 17, 14 : 17) = \begin{pmatrix} 1/6 & 1/24 & 1/12 & 1/24 \\ 1/24 & 1/12 & 1/24 & 0 \\ 1/12 & 1/24 & 1/6 & 1/24 \\ 1/24 & 0 & 1/24 & 1/12 \end{pmatrix}.$$

Ensuite, calculons quelques termes relatifs à l'espace P_2^c , pour cela précisons d'abord les numéros de *ddl* relatifs à cet espace.

Le *ddl* d_5 de E est le premier *ddl* de l'espace d'EF P_2^c , c'est le point de coordonnées $(0, 1)$, le *ddl* d_6 de E est le second *ddl* de l'espace d'EF P_2^c , de coordonnées $(0, 0)$, etc. Pour $i = 1, \dots, 9$, le *ddl* d_{4+i} de E est le i -ème *ddl* du P_2^c .

La fonction de base Φ_5 de E associée au *ddl* d_5 est la fonction de l'espace P_2^c qui vaut 1 en ce point et 0 aux autres *ddl* du P_2^c , la fonction de base Φ_6 de E associée au *ddl* d_6 est la fonction de l'espace P_2^c qui vaut 1 en ce point et 0 aux autres *ddl* du P_2^c , etc. Pour $i = 1, \dots, 9$, la fonction Φ_{4+i} associée au *ddl* d_{4+i} est la fonction de l'espace P_2^c qui vaut 1 en ce *ddl* et 0 aux autres *ddl* du P_2^c .

A titre d'exemple, on calcule les termes de A suivants qui font intervenir Φ_5 , Φ_9 la fonction de base associée au point $(0, 1/2)$ et Φ_{10} la fonction de base associée

au point $(1/2, 1/2)$. On a

$$\begin{aligned}
A(5, 5) &= \int_{K_1} \Phi_5^2 + \int_{K_2} \Phi_5^2 \\
&= \int_{K_1} 4\ell_{1,K_1}^2 (\ell_{1,K_1} - 1/2)^2 + \int_{K_2} 4\ell_{1,K_2}^2 (\ell_{1,K_2} - 1/2)^2 = 1/30, \\
A(9, 9) &= \int_{K_1} \Phi_9^2 = \int_{K_1} 16\ell_{1,K_1}^2 \ell_{2,K_1}^2 = 4/45, \\
A(10, 10) &= \int_{K_1} \Phi_{10}^2 + \int_{K_2} \Phi_{10}^2 \\
&= \int_{K_1} 16\ell_{1,K_1}^2 \ell_{3,K_1}^2 + \int_{K_2} 16\ell_{1,K_2}^2 \ell_{2,K_2}^2 = 8/45, \\
A(5, 9) = A(9, 5) &= \int_{K_1} \Phi_5 \Phi_9 = \int_{K_1} 8\ell_{1,K_1}^2 (\ell_{1,K_1} - 1/2) \ell_{2,K_1} = 0.
\end{aligned}$$

Les autres termes se calculent de manière similaire. On vérifie alors que l'on retrouve bien ces valeurs avec la fonction exemple *exemple1_matrice_elem_T.m*.

Dans cet exemple le calcul de **Ass_Dom** est détaillé pour montrer l'utilisation des fonctions **assemble_coeff_T** (formules de quadrature) et **pre_assemble_T** (valeur des fonctions de base) telles qu'elles sont utilisées par la fonction **matrice_elem_prepa_T**.

Exemple 2 : L'exemple

```
[Ass_Dom, Sc_Dom, A] = exemple2_matrice_elem_T;
```

reprend l'exemple précédent de manière plus condensée en utilisant, la fonction **matrice_elem_prep_T** directement à la place des fonctions **assemble_coeff_T** et **pre_assemble_T**.

C'est aussi un exemple d'utilisation de la fonction **matrice_elem_prep_T** telle qu'elle est utilisée dans la fonction **EF_prepa_don**.

Exemple 3 : L'exemple

```
[Ass_Dom, Sc_Dom, A, Fc_Dom, F] = exemple3_matrice_elem_T;
```

reprend l'exemple précédent en utilisant la fonction **EF_Prep** à la place de la fonction **matrice_elem_prep_T** utilisée dans l'exemple précédent.

Cet exemple est aussi un exemple d'utilisation de la **matrice_elem_prep_T_scm** et assemble la matrice associée à la forme linéaire b associée au second membre avec les valeurs suivantes de f :

$$f = (f_1(x, y), f_2(x, y), f_3(x, y)) = (x^2, x^2 + y^2, x).$$

Le résultat est mis dans le vecteur de type CELL Fc ainsi que dans le vecteur colonne F. Par exemple pour $F(1)$ on trouve

$$\begin{aligned}
 F(1) &= \int_K \Phi_1 x^2 = \int_{K_1} y x^2 + \int_{K_2} (1-x) x^2 = \int_0^1 \left[\int_0^{1-x} y x^2 dy \right] dx \\
 &\quad + \int_0^1 \left[\int_{1-x}^1 (1-x) x^2 dy \right] dx \\
 &= \int_0^1 \frac{1}{2} x^2 (1-x)^2 dx + \int_0^1 (1-x) x^3 dx \\
 &= \frac{2!2!}{2(2+2+1)!} + \frac{1!3!}{(1+3+1)!} = 1/15.
 \end{aligned}$$

Ci-dessus on a utilisé la formule

$$\int_0^1 x^i (1-x)^j dx = \frac{i!j!}{(1+i+j)!}, \forall i, j \in \mathbb{N}.$$

Algorithmme :

EF/Assemblage_elementaire/matrice_elem_T.m

On remet ici la partie déclarative de la fonction :

```

function [Sc_Dom] = matrice_elem_T(donnees, i, T, Tc, X, Y, ...
                                Uc, D, Mat_syst_Dom, Ass_Dom, Sc_Dom) ;

• Ci-dessous ptc{ic} contient les numéros de ddl du ic-ième type d'espace qui
appartiennent au triangle numéro i :
for ic=1 :length(D.E)
    ptc{ic} = Tc{ic}(i, :) ;
end ;

• Coordonnées des 3 sommets du triangle numéro i que l'on note pour la suite a,
b et c de coordonnées (a1, a2), (b1, b2), (c1, c2) :
XT = X(T(i,1 :3)) ;
YT = Y(T(i,1 :3)) ;

• Valeur des composantes, associées aux ddl, de la fonction de l'espace EF produit
E représentée par Uc (Uc représente par exemple l'itéré courant d'une méthode de
point fixe) :
%Boucle sur les espaces de l'espace produit
for ic=1 :length(D.f)
    Uc_pt{ic} = Uc{ic}(ptc{D.f(ic)}) ;

```

end ;

• Calculs de coefficients pour réaliser l'assemblage sur le triangle numéro i . Ces coefficients sont :

- Coordonnées respectives des 3 sommets du triangle : $(a1, a2) = (a_1, a_2)$, $(b1, b2) = (b_1, b_2)$, $(c1, c2) = (c_1, c_2)$,

- Valeur du Jacobien : $de = (a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'$,

- Coordonnées cartésiennes des points de la formule de quadrature : Ac et Bc ,

- Coefficients associés au points de la formule de quadrature : $coeff$,

- Valeurs au points de la formule de quadrature des coordonnées barycentriques relatives à a, b, c données par $L1, L2, L3$:

$[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] \dots$

$= assemble\_coeff\_T(XT, YT, D) ;$

Les points a) b) c) portent sur le calcul des 3 types de termes parmi les 4 types de termes que calcule la fonction `pre_assemble_T` : (le 4-ième type de terme étant la matrice transposée du cas c) :

a) Calcul par type d'espace de numéro $D.E\{ic\}$ des coefficients d'intégration sous forme dupliquée à la dimension de l'espace :

• Boucle sur les types d'espaces utilisés

for $ic=1 : \text{length}(D.E)$

$I\{ic\} = pre\_assemble\_T([], D.E(ic), [], a1, b1, c1, \dots$

$a2, b2, c2, de, Ac, Bc, coeff, 1, D, L1, L2, L3) ;$

end ;

b) Contribution dans l'assemblage de l'équation, du poids de la fonction Uc , qui peut être par exemple l'itéré courant dans une méthode de point fixe :

• Boucle sur les espaces de l'espace produit :

for $ic=1 : \text{length}(D.f)$

$\%Valeurs de l'itere courant aux ddl$

$p1 = Uc\_pt\{ic\} ;$

$\%Type d'espace de l'espace du produit$

$p2 = D.E(D.f(ic)) ;$

$\%Boucle sur les types de derivations [0,1,2]$

for $jc=1 : \text{length}(D.der)$

$uc\{ic,jc\} = pre\_assemble\_T(p1, p2, D.der(jc), a1, b1, \dots$

$c1, a2, b2, c2, de, Ac, Bc, coeff, 2, D, L1, L2, L3) ;$

end ;

end ;

c) Valeurs des fonctions de base et de leur dérivées pour chaque type d'espace à partir de la variable `Ass_Dom` qui, dans le code, a été calculée au préalable par `pre_assemble_T` :

- Boucle sur les espaces de l'espace produit :

```
for ic=1 :length(D.f)
    p1 = D.f(ic) ;
    %boucle sur les types de derivations [0,1,2]
    for jc=1 :length(D.der)
        %Valeurs sur le triangle i de la derivee jc,
        %de fonctions de base de l'espace numero D.E(p1) ;
        sc{ic,jc} = Ass_Dom{i,p1,jc} ;
        %Transposee
        tc{ic,jc} = sc{ic,jc}' ;
    end ;
end ;
```

Les points a) b) c) suivants portent sur l'assemblage de l'équation.

- a) Initialisation des matrices de type cellule décrivant les équations

```
[Mc, Nc] = feval(Mat_syst_Dom, donnees, uc, a1, b1, c1, a2, ...
                b2, c2, de, Ac, Bc, i) ;
```

- b) Assemblage de la première page

```
L2c = assemble_equ(D, I, Mc( :, :,1), Nc( :, :,1), sc, tc) ;
```

- c) Assemblage des autres pages.

```
for k=2 :size(Mc,3)
    L2cp = assemble_equ(D, I, Mc( :, :,k), Nc( :, :,k), sc, tc) ;
    for i1=1 :length(L2c( :)),
        L2c{i1} = L2c{i1} + L2cp{i1} ;
    end ;
end ;
```

- Mise dans la matrice du système :

```
%Boucle sur les espaces de l'espace produit
```

```
for ic=1 :length(D.f)
    p1 = ptc{D.f(ic)} ;
    %Boucle sur les espaces de l'espace produit
```



```

for jc=1 :length(D.f)
    p2 = ptc{D.f(jc)};
    %On n'additionne que les termes non nuls (gain en temps)
    if any(any( L2c{ic,jc} ) )
        Sc_Dom{ic,jc}(p1,p2) = Sc_Dom{ic,jc}(p1,p2) + ...
                                L2c{ic,jc};
    end;
end;
end;
end;

```

Fonction :

```
function [Sc_Fr] = matrice_elem_A(donnees, i, T, Tc, Pf, X, Y, ...
                                Uc, D, Mat_syst_Fr, Ass_Fr, Sc_Fr) ;
```

Calcul de matrices élémentaires pour l'assemblage de formes bilinéaires sur la frontière associées à l'EDP. C'est la partie en a_Γ de la forme bilinéaire $\mathcal{A}$ dans (1.5).

Cette fonction est analogue à la fonction `matrice_elem_T`, avec une variable d'entrée supplémentaire, la variable `Pf` (cf. Tableau 2.2.3-1b) qui décrit les arêtes de la frontière suivant une numérotation, localement sur la frontière, des arêtes. La variable `i`, que l'on notera aussi i , indique dans cette numérotation le numéro d'arête sur laquelle on intègre.

Pour une arête frontière donnée, désignée ci-dessus par son numéro i et située sur le triangle $\text{Pf}(i, 1)$, la commande

```
[Sc_Fr] = matrice_elem_A(donnees, i, T, Tc, Pf, X, Y, ...
                        Uc, D, Mat_syst_Fr, Ass_Fr, Sc_Fr) ;
```

permet de calculer la matrice élémentaire relative à la forme bilinéaire associée au système d'EDP et aux termes d'arêtes frontières. La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP et aux arêtes frontières (désigné ci-dessus par `Sc_Fr`, matrice de type CELL).

Variables d'entrée (qui diffèrent de celles de `matrice_elem_T`) :

Les variables sont identiques à celles de la fonction `matrice_elem_T` excepté les variables décrites ci-dessous : `i` et aussi `Mat_syst_Fr`, `Ass_Fr` et `Sc_Fr` successivement à la place de `Mat_syst_Dom`, `Ass_Dom`, `Sc_Dom` dans `matrice_elem_T` et enfin la variable `Pf`. Précisons ces variables.

i : DOUBLE ARRAY. Scalaire désignant le numéro d'arête, dans la numérotation locale des arêtes sur la frontière.

Pf : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes sur la frontière) $\times$ 4. Sur la frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur la frontière. Soit une arête de numéro j dans sa numérotation sur le morceau la frontière, $\text{Pf}(j, 1)$ est le numéro de triangle auquel appartient cette arête, $\text{Pf}(j, 2)$ son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3), $\text{Pf}(j, 3)$ son numéro global d'arête dans la numérotation globale des arêtes et $\text{Pf}(j, 4)$ le numéro de morceau de frontière auquel appartient cette arête. `Pfc` vaut [] si `Front` = []. Les variables `Pf` et `Pfc` contiennent la même information si l'on sait que $\text{Pf}(:, 1 : 3) = [\text{Pfc}\{1\}; \dots; \text{Pfc}\{\text{end}\}]$. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Mat_syst_Fr : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur Γ et qui correspond au terme a_Γ qui intervient

dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce `handle` correspond à la valeur du champ `freq_Fr` du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a `mat_syst_Fr = donnees.freq_Fr`. La syntaxe de la fonction désignée par `Mat_syst_Fr`, que nous notons ci-dessous `eq_Fr`, est la suivante :

Syntaxe de la fonction appelée par `matrice_elem_A`.

```
function [Mc, Nc] = eq_Fr(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

avec :

Variables d'entrée (pour `eq_Fr`) :

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF.

numF : DOUBLE ARRAY. Scalaire désignant le numéro de frontière courante pour les fonctions d'arêtes : on a $\text{numF} = \text{Pf}(i, 1)$ avec i numéro d'arête, dans la numérotation locale des arêtes sur la frontière

uc : CELL ARRAY. Tableau $n \times 3$ de cellules. Pour $i = 1, \dots, n$ ($n = \dim(E)$ est le nombre d'inconnues) et $j = 1, 2, 3$ (j indique la dérivation $j - 1$), $\text{uc}\{i, j\}$ est un vecteur ligne de type double array. Pour $j = 0, 1, 2$, le terme $\text{uc}\{i, j + 1\}$ représente les valeurs, aux points de la formule de quadrature utilisée pour l'intégration sur les 3 arêtes du triangle qui porte l'arête frontière courante, de la dérivée d'ordre $j = 0, 1, 2$ de la i -ème composante de l'itéré courant X^0 de (1.5) (et (1.10) pour le second membre), représenté ici par le vecteur Uc . Ces valeurs sont données par $X_{i,j}^0(\text{Ac}, \text{Bc})$. Cette variable est décrite au Tableau 2.2.4-1, première partie, du Paragraphe 2.2.4.

Dans la fonction `matrice_elem_A`, pour l'intégration qui est effectuée uniquement sur l'arête frontière, le vecteur ligne de trois termes `c` qui définit le poids de l'intégration sur chaque arête prendra alors la valeur nulle sauf pour le terme `numALoc` avec $c(\text{numALoc}) = 1$.

a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Scalars désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c du triangle K de numéro `numK`.

de : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature. Il s'agit ici de la formule de quadrature de numéro donné par **D.intA**.

numK : DOUBLE ARRAY. Scalaire, désigne le numéro de triangle courant auquel appartiennent les points de coordonnées cartésiennes **Ac** et **Bc**. On a ici **numK** = **Pf**($i, 1$), avec i numéro d'arête, dans la numérotation locale des arêtes sur la frontière.

numALoc : DOUBLE ARRAY. Scalaire désignant le numéro local sur le triangle **numK** : on a **numALoc** = **Pf**($i, 2$) avec i numéro d'arête, dans la numérotation locale des arêtes sur la frontière et **numK** = **Pf**($i, 1$).

numA : DOUBLE ARRAY. Scalaire désignant le numéro global de l'arête courante : on a **numA** = **Pf**($i, 3$) avec i numéro d'arête, dans la numérotation locale des arêtes sur la frontière.

Variable de sortie (pour eq_Fr) :

Mc, Nc : CELL ARRAY. Les variables de sortie **Mc** et **Nc** sont les variables de type **cell** décrivant l'EDP sur le domaine. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de matrice_elem_A :

Ass_Fr : CELL ARRAY. Tableau de cellules $N \times N_E \times 3$ où N est le nombre de d'arêtes sur la frontière et $N_E = \text{length}(\text{donnees.D.E})$. Sur la frontière, les arêtes reçoivent une numérotation locale, l'arête numéro i est décrite par **Pf**($i, :$). L'ordre choisi est celui des arêtes de **Front**{1}, **Front**{2} etc. (cf. aussi les variables **Pf** et **Pfc**). Le terme **Ass_Fr**{ i, j, k } est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K de numéro **Pf**($i, 1$) qui porte l'arête de numéro i dans la numérotation locale des arêtes sur la frontière, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par **donnees.D.E**(j). Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.intA** utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace d'EF sur le triangle, l'ordre des fonctions de base étant celui donné au Paragraphe 2.3.3. Cette variable représente le champ **Ass_Fr** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Sc_Fr : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Représentation de la matrice par blocs correspondant à l'assemblage de la partie en Γ de la forme bilinéaire $\mathcal{A}$ (terme en a_Γ) de (1.5), Paragraphe 1.2. Analogie de **Sc_Dom**, ici, pour l'assemblage de la partie en Γ au lieu de Ω . La forme bilinéaire est définie par le champ **donnees.feq_Fr**. L'assemblage se fait

par boucle sur les arêtes de la frontière, le rôle de la fonction `matrice_elem_A` est de calculer la contribution de l'arête frontière courante à la forme bilinéaire. Le terme `Sc_Fr` $\{i, j\}$ est la matrice de taille $\dim(E_i) \times \dim(E_j)$ qui correspond à la matrice d'assemblage de la restriction de la partie de la forme bilinéaire qui porte sur $E_j \times E_i$, E_k désignant le k -ième espace de l'espace EF produit E et de numéro d'espace `donnees.D.E(donnees.D.f(k))`.

Variables de sortie :

`Sc_Fr` : CELL ARRAY. Matrice en entrée de la fonction augmentée de la contribution de a_Γ .

Exemple : Cet exemple est pour la fonction `matrice_elem_A` l'analogue de l'exemple 3 pour la fonction `matrice_elem_T`, la seule différence est que les équations sont posées sur la frontière Γ ainsi qu'en un point du domaine (pour obtenir l'unicité de la solution) au lieu de l'être sur le domaine Ω .

On reprend le maillage et l'espace E utilisés dans les 3 exemples précédents pour la fonction `matrice_elem_T` ainsi que le second membre utilisé dans le troisième de ces exemples. On considère donc le maillage du carré par les deux triangles de la figure 4.3-10, l'espace EF produit $E = P_1^c \times P_2^c \times P_1^c$ muni de sa base nodale $\{\Phi_k\}_{1 \leq k \leq 17}$ ainsi que la fonction second membre

$$f = (f_1(x, y), f_2(x, y), f_3(x, y)) = (x^2, x^2 + y^2, x).$$

Dans cet exemple, on résout sur la frontière Γ du carré unité et au point milieu du carré le système :

$$\begin{aligned} \exists u = (u_1, u_2, u_3) \text{ tel que :} \\ u_1 = f_1, u_2 = f_2, u_3 = f_3 \text{ sur } \Gamma, \\ u_2(0.5, 0.5) = f_2(0.5, 0.5) = 0.5^2 + 0.5^2 = 0.5. \end{aligned}$$

On considère alors le problème approché sous forme variationnelle :

$\exists u = (u_1, u_2, u_3) \in E$ tel que $\forall v = (v_1, v_2, v_3) \in E$ on a :

$$\int_{\Gamma} u_1 v_1 = \int_{\Gamma} f_2 v_1 ds, \int_{\Gamma} u_2 v_2 = \int_{\Gamma} f_2 v_2 ds, \int_{\Gamma} u_3 v_3 = \int_{\Gamma} f_3 v_3 ds.$$

Ce problème se met sous la forme équivalente suivante :

$$\begin{aligned} \exists u = (u_1, u_2, u_3) \in E \text{ tel que } a(u, v) = b(v), \forall v = (v_1, v_2, v_3) \in E \\ u_2(0.5, 0.5) = 0.5, \end{aligned}$$

avec a la forme bilinéaire définie sur $E \times E$ par

$$a(u, v) = \int_{\Gamma} u_1 v_1 ds + \int_{\Gamma} u_2 v_2 ds + \int_{\Gamma} u_3 v_3 ds, \forall (u, v) \in E \times E$$

et b la forme linéaire définie sur E par

$$b(v) = \int_{\Gamma} f_1 v_1 ds + \int_{\Gamma} f_2 v_2 ds + \int_{\Gamma} f_3 v_3 ds, \forall v \in E.$$

Le problème EF est alors bien posé car $u \rightarrow [\int_{\Gamma} u^2 ds]^{1/2}$ est une norme sur l'espace d'EF P_1^c et $u \rightarrow [\int_{\Gamma} u^2 ds + u(0.5, 0.5)^2]^{1/2}$ une norme sur l'espace d'EF P_2^c .

Dans l'exemple qui suit on calcule la matrice A du système associé à a et à la condition ponctuelle ainsi que le vecteur second membre F associé à b et à la condition ponctuelle. Pour assembler le système on calcule d'abord la matrice et le second membre associés à la formulation variationnelle de la première équation (sans la condition ponctuelle). La fonction de base Φ_{10} , appartenant à l'espace P_2^c , associée au *ddl* de coordonnées $(0.5, 0.5)$ étant nulle sur le bord du carré, la ligne et la colonne 10 de la matrice ainsi que la ligne 10 du second membre sont donc nuls. Pour prendre en compte la condition ponctuelle, on pose alors $A(10, 10) = 1$ et $F(10) = f_2(m) = 1/2$. On vérifie que le système obtenu est bien équivalent au problème variationnel ci-dessus.

La commande

```
[Ass_Fr, Sc_Fr, A, Fc_Fr, F] = exemple3_matrice_elem_A ;
```

résout alors le problème EF. Le fichier reprend celui de l'exemple 3 précédent dans la description de la fonction `matrice_elem_T` en l'adaptant au cas des fonctions `matrice_elem_A` et `matrice_elem_A_scm`.

Calculons quelques valeurs de la matrice du système A que l'on comparera, à titre de vérification, avec la matrice calculée par la commande ci-dessus. Pour détailler les calculs on a représenté à la figure 4.3-12 la numérotation des arêtes frontières pour le maillage utilisé.

Numéros de sommets et d'arêtes frontières

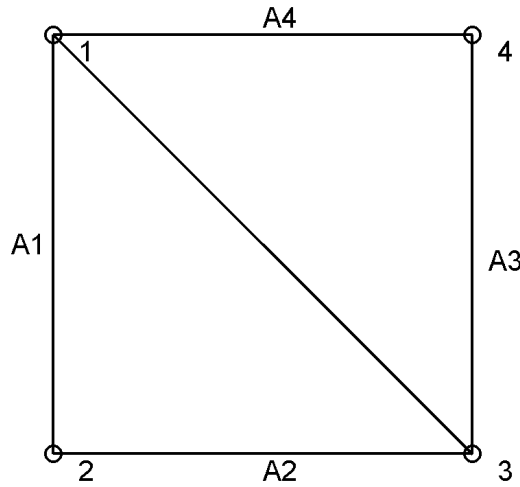


Figure 4.3-12. Numérotation, localement, des arêtes frontières.

Ci-dessous Φ_1, Φ_2, Φ_3 et Φ_4 sont les fonctions de bases du P_1^c attachées respectivement au sommets numéros 1, 2, 3 et 4. Ensuite Φ_5 et Φ_6 sont les fonctions de bases du P_2^c attachées respectivement au sommets numéros 1 et 2. Et enfin Φ_9 et Φ_{10} sont les fonctions de bases du P_2^c attachées respectivement aux sommets milieux de l'arête de sommets 1 et 2 et de l'arête 1 et 3 (milieu du carré). On a

$$\begin{aligned}
A(1, 1) &= \int_{\Gamma} \Phi_1^2 ds = \int_{A_1} \Phi_1^2 ds + \int_{A_4} \Phi_1^2 ds = \int_0^1 y^2 dy + \int_0^1 (1-x)^2 dx = 2/3, \\
A(1, 2) &= A(2, 1) = \int_{\Gamma} \Phi_1 \Phi_2 ds = \int_{A_1} \Phi_1 \Phi_2 ds = \int_0^1 y(1-y) dx = 1/6, \\
A(2, 2) &= \int_{\Gamma} \Phi_2^2 ds = \int_{A_1} \Phi_2^2 ds + \int_{A_2} \Phi_2^2 ds = \int_0^1 (1-y)^2 dy + \int_0^1 (1-x)^2 dx = 2/3, \\
A(5, 5) &= \int_{\Gamma} \Phi_5^2 ds = \int_{A_1} \Phi_5^2 ds + \int_{A_4} \Phi_5^2 ds \\
&= \int_0^1 2^2 y^2 (y-1/2)^2 dy + \int_0^1 2^2 (1-x)^2 (1/2-x)^2 dx = 4/15, \\
A(5, 6) &= A(6, 5) = \int_{\Gamma} \Phi_5 \Phi_6 ds = \int_{A_1} \Phi_5 \Phi_6 ds \\
&= \int_0^1 4y(y-1/2)(1-y)(1/2-y) dy = -1/30, \\
A(6, 6) &= \int_{\Gamma} \Phi_6^2 ds = \int_{A_1} \Phi_6^2 ds + \int_{A_2} \Phi_6^2 ds \\
&= \int_0^1 2^2 (1-y)^2 (1/2-y)^2 dy + \int_0^1 2^2 (1-x)^2 (1/2-x)^2 dx = 4/15, \\
A(9, 9) &= \int_{\Gamma} \Phi_9^2 ds = \int_{A_1} \Phi_9^2 ds = \int_0^1 4^2 y^2 (1-y)^2 dy = 8/15, \\
A(5, 9) &= A(9, 5) = \int_{\Gamma} \Phi_5 \Phi_9 ds = \int_{A_1} \Phi_5 \Phi_9 ds \\
&= \int_0^1 8y(y-1/2)y(1-y) dy = A(6, 9) = A(9, 6) = 1/15 \\
A(10, :) &= A(:, 10) = 0 \text{ pour la forme bilinéaire du système puis } A(10, 10) = 1.
\end{aligned}$$

Les calculs des autres termes se font de manière analogue. Pour le second membre on trouve, pour les premiers termes :

$$\begin{aligned}
F(1) &= \int_{\Gamma} x^2 \Phi_1 ds = \int_{A_1} x^2 \Phi_1 ds + \int_{A_4} x^2 \Phi_1 ds \\
&= \int_0^1 0^2 \times y dy + \int_0^1 x^2 \times (1-x) dx = 1/12,
\end{aligned}$$

$$\begin{aligned}
F(2) &= \int_{\Gamma} x^2 \Phi_2 ds = \int_{A_1} x^2 \Phi_2 ds + \int_{A_2} x^2 \Phi_2 ds \\
&= \int_0^1 0^2 \times (1-y) dy + \int_0^1 x^2 \times (1-x) dx = 1/12, \\
F(3) &= \int_{\Gamma} x^2 \Phi_3 ds = \int_{A_2} x^2 \Phi_3 ds + \int_{A_3} x^2 \Phi_3 ds \\
&= \int_0^1 x^2 \times x dx + \int_0^1 1^2 \times (1-y) dy = 3/4, \\
F(4) &= \int_{\Gamma} x^2 \Phi_4 ds = \int_{A_3} x^2 \Phi_4 ds + \int_{A_4} x^2 \Phi_4 ds \\
&= \int_0^1 1^2 \times y dy + \int_0^1 x^2 \times x dx = 3/4, \\
F(5) &= \int_{\Gamma} (x^2 + y^2) \Phi_5 ds = \int_{A_1} (x^2 + y^2) \Phi_5 ds + \int_{A_4} (x^2 + y^2) \Phi_5 ds \\
&= \int_0^1 (0^2 + y^2) \times 2y(y - 1/2) dy + \int_0^1 (x^2 + 1^2) \times 2(1-x)(1/2 - x) dx \\
&= 3/10. \\
F(6) &= \int_{\Gamma} (x^2 + y^2) \Phi_6 ds = \int_{A_1} (x^2 + y^2) \Phi_6 ds + \int_{A_2} (x^2 + y^2) \Phi_6 ds \\
&= \int_0^1 (0^2 + y^2) \times 2(1-y)(1/2 - y) dy + \int_0^1 (x^2 + 0^2) \times 2(1-x)(1/2 - x) dx \\
&= -1/30.
\end{aligned}$$

etc.

Algorithme :

EF/Assemblage_elementaire/matrice_elem_A.m

On remet ici la partie déclarative de la fonction :

```
function [Sc_Fr] = matrice_elem_A(donnees, i, T, Tc, Pf, X, Y,...
                                Uc, D, Mat_syst_Fr, Ass_Fr, Sc_Fr) ;
```

- Numéro du triangle auquel appartient l'arête i :

```
numK = Pf(i,1) ;
```

- Numéro localement de l'arête i sur le triangle :

```
numALoc = Pf(i,2) ;
```

- Numéro global de l'arête i :

```
numA = Pf(i,3) ;
```

- Numéro de frontière à laquelle appartient l'arête i :


```
numF = Pf(i,4) ;
```

• Ci-dessous `ptc{ic}` contient les numéros de *ddl* du *ic-ième* type d'espace qui appartiennent au triangle numéro `numK` :

```
for ic=1 :length(D.E)
```

```
    ptc{ic} = Tc{ic}(numK, :) ;
```

```
end ;
```

• Coordonnées des 3 sommets du triangle numéro `numK` que l'on note pour la suite *a*, *b* et *c* de coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) :

```
XT = X(T(numK,1 :3)) ;
```

```
YT = Y(T(numK,1 :3)) ;
```

• Valeur des composantes, associées aux *ddl*, de la fonction de l'espace EF produit *E* représentée par `Uc` (`Uc` représente par exemple l'itéré courant d'une méthode de point fixe) :

```
%Boucle sur les espaces de l'espace produit
```

```
for ic=1 :length(D.f)
```

```
    Uc_ptc{ic} = Uc{ic}(ptc{D.f(ic)}) ;
```

```
end ;
```

• Le poids est mis sur l'arête courante :

```
c = [0,0,0] ;
```

```
c(numALoc) = 1 ;
```

• Calculs de coefficients pour réaliser l'assemblage sur le triangle numéro *i*. Ces coefficients sont :

- Coordonnées respectives des 3 sommets du triangle : $(a1, a2) = (a_1, a_2)$, $(b1, b2) = (b_1, b_2)$, $(c1, c2) = (c_1, c_2)$,

- Valeur du Jacobien : $de = (a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'$,

- Coordonnées cartésiennes des points de la formule de quadrature : *Ac* et *Bc*,

- Coefficients associés au points de la formule de quadrature : *coeff*,

- Valeurs au points de la formule de quadrature des coordonnées barycentriques relatives à *a*, *b*, *c* données par *L1*, *L2*, *L3* :

```
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
```

```
assemble_coeff_A(XT, YT, c, D) ;
```

Les points a) b) c) portent sur le calcul des 3 types de termes parmi les 4 types de termes que calcule la fonction `pre_assemble_A` : (le 4-ième type de terme étant la matrice transposée du cas c) :

a) Calcul par type d'espace de numéro `D.E{ic}` des coefficients d'intégration sous forme dupliquée à la dimension de l'espace :

- Boucle sur les types d'espaces utilisés

```
for ic=1 :length(D.E)
    I{ic} = pre_assemble_A([],D.E(ic), [], a1, b1, c1, ...
        a2, b2, c2, de, Ac, Bc, coeff, 1, D, L1, L2, L3);
end;
```

b) Contribution dans l'assemblage de l'équation, du poids de la fonction U_c , qui peut être par exemple l'itéré courant dans une méthode de point fixe :

- Boucle sur les espaces de l'espace produit :

```
%boucle sur les espaces de l'espace produit
for ic=1 :length(D.f)
    %boucle sur les types de derivation
    for jc=1 :length(D.der)
        uc{ic,jc} = pre_assemble_A(Uc_pt{ic}, D.E(D.f(ic)), ...
            D.der(jc), a1, b1, c1, a2, b2, c2, de, Ac, Bc, ...
                coeff, 2, D, L1, L2, L3);
    end;
end;
```

c) Valeurs des fonctions de base et de leur dérivées pour chaque type d'espace à partir de la variable `Ass_Fr` qui, dans le code, a été calculée au préalable par `pre_assemble_A` :

- Boucle sur les espaces de l'espace produit :

```
%boucle sur les espaces de l'espace produit
for ic=1 :length(D.f)
    %boucle sur les types de derivation
    for jc=1 :length(D.der)
        sc{ic,jc} = Ass_Fr{i,D.f(ic),jc};
        tc{ic,jc} = sc{ic,jc}';
    end;
end;
```

Les points a) b) c) suivants portent sur l'assemblage de l'équation.

- a) Initialisation des matrices de type cellule décrivant les équations

```
[Mc, Nc] = feval(Mat_syst_Fr, donnees, numF, uc, a1, b1, ...
    c1, a2, b2, c2, de, Ac, Bc, numK, numALoc, numA);
```

- b) Assemblage de la première page

```
L2c = assemble_equ(D, I, Mc(:, :,1), Nc(:, :,1), sc, tc);
```

- c) Assemblage des autres pages.

```

for k=2 :size(Mc,3)
    L2cp = assemble_equ(D, I, Mc( :, :,k), Nc( :, :,k), sc, tc);
    for i1=1 :length(L2c( :)),
        L2c{i1} = L2c{i1} + L2cp{i1};
    end;
end;

• Mise dans la matrice du système :

%Boucle sur les espaces de l'espace produit :
for ic=1 :length(D.f)
    %Boucle sur les espaces de l'espace produit :
    for jc=1 :length(D.f)
        Sc_Fr{ic,jc}(ptc{D.f(ic)},ptc{D.f(jc)}) = ...
            Sc_Fr{ic,jc}(ptc{D.f(ic)},ptc{D.f(jc)}) + L2c{ic,jc};
    end
end;
end;

```

Fonction :

```
function [Sc_Ad] = matrice_elem_Ad(donnees, i, T, Tc, ...
                                   Pa, X, Y, Uc, D, Mat_syst_Ad, Ass_Ad, Sc_Ad) ;
```

En développement, fonctionne. Calcul de matrices élémentaires pour l'assemblage de forme bilinéaires sur les arêtes du maillage associées à l'EDP. C'est la partie en $a_{\bar{\Sigma}}$ de la forme bilinéaire $\mathcal{A}$ dans (1.5).

Cette fonction est analogue à la fonction `matrice_elem_T`, avec une variable d'entrée supplémentaire, la variable `Pa` (cf. Tableau 2.2.3-1b) qui décrit les arêtes du maillage dans la numérotation globale des arêtes du domaine. La variable `i`, que l'on notera aussi i , indique dans cette numérotation le numéro d'arête sur laquelle on intègre.

L'intégration se fait sur la frontière Γ et sur Σ la réunion des arêtes intérieures. Pour tenir compte de l'intégration de fonctions discontinues, on définit l'intégrale sur Σ par

$$\int_{\Sigma} f = \frac{1}{2} \int_{\Sigma_1} f + \frac{1}{2} \int_{\Sigma_2} f$$

où on note Σ_p , $p = 1, 2$ la réunion des arêtes intérieures, avec la convention que sur une arête i de Σ_1 (resp. Σ_2) on intègre la restriction de f au triangle de numéro $\text{Pa}(i, 1)$ (resp. $\text{Pa}(i, 3)$). Pour plus de détails on pourra consulter aussi le `help` de cette fonction.

Pour une arête donnée, désignée ci-dessus par son numéro i et située sur le triangle $\text{Pa}(i, 1)$ ou $\text{Pa}(i, 3)$, la commande

```
[Sc_Ad] = matrice_elem_Ad(donnees, i, T, Tc, ...
                           Pa, X, Y, Uc, D, Mat_syst_Ad, Ass_Ad, Sc_Ad) ;
```

permet de calculer la matrice élémentaire relative à la forme bilinéaire associée au système d'EDP et aux termes d'arêtes. La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP et aux arêtes, désigné ci-dessus par `Sc_Ad`, matrice de type CELL.

On note que comme dans la fonction `matrice_elem_A` vue précédemment, il y a ici aussi une prise en compte d'arêtes sur la frontières.

Variables d'entrée (qui diffèrent de celles de `matrice_elem_T`) :

Les variables sont identiques à celles de la fonction `matrice_elem_T` excepté les variables décrites ci-dessous : `i` et aussi `Mat_syst_Ad`, `Ass_Ad` et `Sc_Ad` successivement à la place de `Mat_syst_Dom`, `Ass_Dom`, `Sc_Dom` dans `matrice_elem_T` et enfin la variable `Pa`. Précisons ces variables.

`i` : DOUBLE ARRAY. Scalaire désignant le numéro d'arête, dans la numérotation globale des arêtes du maillage, noté aussi i ci-dessous.

Pa : DOUBLE ARRAY. Matrice de taille (Nombre d'arête du maillage) $\times$ 5. Description des arêtes du maillage tel que **Pa**(j , 1) est un numéro de triangle auquel appartient l'arête j et **Pa**(j , 2) son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour **Pa**(j , 3 : 4) si l'arête appartient à un deuxième triangle sinon **Pa**(j , 3) et **Pa**(j , 4) sont nuls (arête sur la frontière). Enfin **Pa**(j , 5) est le numéro du morceau de frontière si l'arête j est sur la frontière, nul sinon ou si on a posé **Front** = []. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Mat_syst_Ad : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur $\bar{\Sigma}$ et qui correspond au terme $a_{\bar{\Sigma}}$ qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feq_Ad** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_syst_Ad** = **donnees.feq_Ad**. En développement, fonctionne. La syntaxe de la fonction désignée par **Mat_syst_Ad**, que nous notons ci-dessous **eq_Ad**, est analogue pour les variables à celle de la fonction appelée par la fonction **matrice_elem_A**, sa syntaxe est la suivante :

Syntaxe de la fonction appelée par matrice_elem_Ad.

```
function [Mc, Nc] = eq_Ad(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

avec :

Variables d'entrée (pour **eq_Ad**) :

Le terme **numA** est le numéro de l'arête courante donné par la valeur de i , dans la numérotation globale des arêtes du maillage. Le terme **numK** peut alors prendre la valeur **numK** = **Pa**(i , 1) qui est la valeur du numéro du premier triangle auquel appartient cette arête ou bien **Pa**(i , 3) si celle-ci appartient à un second (éventuel) triangle. De même **numALoc** prendra alors respectivement les valeurs **numALoc** = **Pa**(i , 2) ou bien **numALoc** = **Pa**(i , 4).

donnees : STRUCT ARRAY. Variable décrivant les données du problème EF.

numF : DOUBLE ARRAY. Scalaire désignant le numéro de frontière courante pour les fonctions d'arêtes : on a **numF** = **Pa**(i , 5) avec i numéro d'arête, dans la numérotation globale des arêtes du maillage. Pour une arête intérieure **numF** = 0.

uc : CELL ARRAY. Tableau $n \times 3$ de cellules. Pour $i = 1, \dots, n$ ($n = \dim(E)$ est le nombre d'inconnues) et $j = 1, 2, 3$ (j indique la dérivation $j - 1$) , **uc**{ i , j } est un vecteur ligne de type double array. Pour $j = 0, 1, 2$, le terme **uc**{ i , $j + 1$ } représente les valeurs, aux points de la formule de quadrature utilisée pour l'intégration sur les 3 arêtes du premier ou du second (éventuel) triangle de numéro **numK** qui porte l'arête courante, de la dérivée d'ordre $j = 0, 1, 2$ de la i -ème composante de l'itéré courant X^0 de (1.5) (et (1.10) pour le second membre), représenté ici par le vecteur **Uc**. Ces valeurs sont données par $X_{i,j}^0(\mathbf{Ac}, \mathbf{Bc})$. Cette variable est décrite au Tableau 2.2.4-1, première partie, du Paragraphe 2.2.4.

Dans la fonction `matrice_elem_Ad`, pour l'intégration qui est effectuée uniquement sur l'arête courante, le vecteur ligne de trois termes `c` qui définit le poids de l'intégration sur chaque arête prendra alors la valeur nulle sauf pour le terme `numALoc` avec $c(\text{numALoc}) = 1$.

`a1, b1, c1, a2, b2, c2` : DOUBLE ARRAY. Scalaires désignant les coordonnées (a_1, a_2) , (b_1, b_2) et (c_1, c_2) des sommets a, b, c du triangle K de numéro `numK`.

`de` : DOUBLE ARRAY. Scalaire désignant le déterminant de la matrice jacobienne de la bijection affine du triangle de référence $\hat{K}$ de sommets $(0, 0)$, $(1, 0)$ et $(0, 1)$ sur le triangle de sommets a, b, c de coordonnées $a = (a_1, a_2)$, $b = (b_1, b_2)$ et $c = (c_1, c_2)$ et qui a pour valeur :

$$(a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'.$$

Rappelons que cette valeur représente aussi un produit vectoriel de valeur absolue égale au double de la surface du triangle K (surface du parallélogramme construit sur a, b, c).

`Ac, Bc` : DOUBLE ARRAY. Vecteurs ligne des coordonnées cartésiennes en x et en y des points de la formule de quadrature. Il s'agit ici de la formule de quadrature de numéro donné par `D.intA`.

`numK` : DOUBLE ARRAY. Scalaire, désigne le numéro de triangle courant auquel appartiennent les points de coordonnées cartésiennes `Ac` et `Bc`. Le terme `numK` peut prendre la valeur `numK = Pa(i, 1)` ou la valeur `numK = Pa(i, 3)` (s'il existe un second triangle auquel appartient l'arête courante) avec i numéro d'arête, dans la numérotation globale des arêtes du maillage.

`numALoc` : DOUBLE ARRAY. Scalaire désignant le numéro local de l'arête numéro `numA` sur le triangle de numéro `numK` avec `numA = i`, numéro d'arête dans la numérotation globale des arêtes du maillage et avec `numK = Pa(i, 1)` ou bien `numK = Pa(i, 3)`, on a alors, suivant le cas, `numALoc = Pa(i, 2)` ou bien `numALoc = Pa(i, 4)`.

`numA` : DOUBLE ARRAY. Scalaire désignant le numéro global de l'arête courante : on a `numA = i` avec i numéro d'arête, dans la numérotation globale des arêtes du maillage.

Variable de sortie (pour `eq_Ad`) :

`Mc, Nc` : CELL ARRAY. Les variables de sortie `Mc` et `Nc` sont les variables de type `cell` décrivant l'EDP sur le domaine. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de `matrice_elem_Ad` :

Ass_Ad : CELL ARRAY. Tableau de cellules $N \times N_E \times 3 \times 2$ où N est le nombre d'arêtes et $N_E = \text{length}(\text{donnees.D.E})$. Le terme **Ass_Ad** $\{i, j, k, m\}$ est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K_m de numéro $Pa(i, 1)$ pour $m = 1$ et $Pa(i, 3)$ pour $m = 2$ pour l'arête de numéro i dans la numérotation globale des arêtes, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par **donnees.D.E**(j). Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Pour $m = 1$, on obtient le résultat concernant la restriction des fonctions de base sur le premier triangle auquel appartient l'arête i et de numéro donné par **Pa**($i, 1$), pour $m = 2$, le résultat concerne l'autre éventuel triangle de numéro **Pa**($i, 3$), s'il n'y a pas de second triangle on pose **Ass_Dom** $\{i, j, k, m\} = []$. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro **donnees.intA** utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ **Ass_Ad** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Sc_Ad : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Analogue de **Sc_Fr** pour l'assemblage dans la forme bilinéaire $\mathcal{A}$ de (1.5), Paragraphe 1.2, de la partie en $\bar{\Sigma}$ (réunion de toutes les arêtes du maillage) avec l'intégration du terme en $a_{\bar{\Sigma}}$. A ce terme s'ajoute la contribution du terme en a_{LR} de la méthode de Lesaint-Raviart. La forme bilinéaire $\bar{\Sigma}$ est définie par le champ **donnees.feq_Ad** et l'assemblage se fait par la fonction **matrice_elem_Ad**. Pour la méthode de Lesaint-Raviart, la forme bilinéaire est définie par le champ **donnees.D.LR** et l'assemblage se fait par la fonction **matrice_elem_Ad_int_LR**. La partie en $a_{\bar{\Sigma}}$ pourra être modifiée par l'utilisateur suivant le problème à traiter, ceci par le biais d'une fonction analogue à celle de **matrice_elem_Ad** à insérer dans une boucle du même type que celle effectuée sur les arêtes dans la fonction **EF_assemblage**.

Mat_syst_Ad : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur $\bar{\Sigma}$ et qui correspond au terme $a_{\bar{\Sigma}}$ qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). Ce **handle** correspond à la valeur du champ **feof_Ad** du Tableau 2.2.1-1b du Paragraphe 2.2.1. En pratique dans le code on a **mat_syst_Ad** = **donnees.feq_Ad**. En développement, fonctionne.

Variables de sortie :

Sc_Ad : CELL ARRAY. Matrice en entrée de la fonction augmentée de la contribution de $a_{\bar{\Sigma}}$.

Exemple 1. Cet exemple reprend l'exemple précédent vu dans la description de la fonction **matrice_elem_A** où l'équation est remplacée ici par :

$$u_1 = x^2, u_2 = x^2 + y^2, u_3 = x \text{ sur } \bar{\Sigma} = \Gamma \cup \Sigma.$$

c'est donc seulement le domaine de validité de l'équation qui change. Les arêtes de Γ et Σ ont représentées à la figure 4.3-13 suivante avec leur numérotation globale d'arêtes.

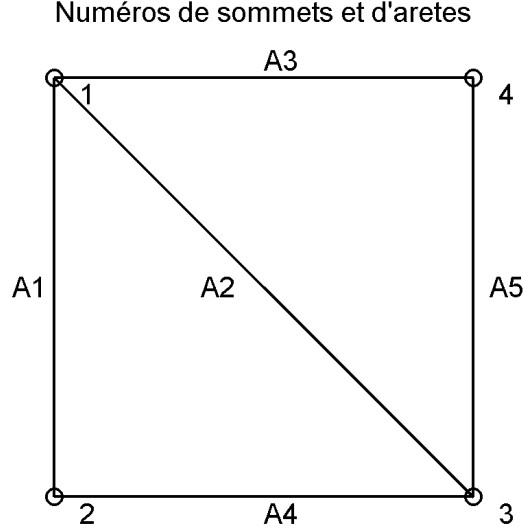


Figure 4.3-13. Numérotation globale des arêtes.

On a ainsi avec ce maillage $\Gamma = A_1 \cup A_3 \cup A_4 \cup A_5$ et $\Sigma = A_2$ est réduit à une arête. On remarque que la numérotation globale des arêtes situées sur la frontière diffère en général de leur numérotation locale sur la frontière (*cf.* Figure 4.3-12).

Le problème se met sous la formulation variationnelle suivante :

$\exists u = (u_1, u_2, u_3) \in E$ tel que $\forall v = (v_1, v_2, v_3) \in E$ on a :

$$\int_{\Gamma \cup \Sigma} u_1 v_1 = \int_{\Gamma \cup \Sigma} f_1 v_1, \int_{\Gamma \cup \Sigma} u_2 v_2 = \int_{\Gamma \cup \Sigma} f_2 v_2, \int_{\Gamma \cup \Sigma} u_3 v_3 = \int_{\Gamma \cup \Sigma} f_3 v_3,$$

ou encore

$$\exists u = (u_1, u_2, u_3) \in E \text{ tel que } a(u, v) = b(v), \forall v = (v_1, v_2, v_3) \in E$$

avec a la forme bilinéaire définie sur $E \times E$ par

$$a(u, v) = \int_{\Gamma \cup \Sigma} u_1 v_1 + \int_{\Gamma \cup \Sigma} u_2 v_2 + \int_{\Gamma \cup \Sigma} u_3 v_3, \forall (u, v) \in E \times E$$

et b la forme linéaire définie sur E par

$$b(v) = \int_{\Gamma \cup \Sigma} f_1 v_1 + \int_{\Gamma} f_2 v_2 + \int_{\Gamma \cup \Sigma} f_3 v_3, \forall v \in E.$$

Le problème est alors bien posé car $u \rightarrow \sqrt{a(u, u)}$ est une norme sur E .

La commande

`[Ass_Ad, Sc_Ad, A, Fc_Ad, F] = exemple3_matrice_elem_Ad ;`

résout alors le problème EF. Le fichier reprend celui de l'exemple précédent du cas de la fonction `matrice_elem_A` adapté au cas des fonctions `matrice_elem_Ad` et `matrice_elem_Ad_scm`.

Dans le calcul de A et F , viennent s'ajouter par rapport aux matrices vues à l'exemple précédent la contribution de l'arête A_2 . Donnons quelques exemples de calcul de ces deux matrices.

Soit Φ_1 la fonction de base P_1^c de E qui vaut 1 au sommet de coordonnées $(0, 1)$ et 0 aux autres ddl du P_1^c . On a ainsi

$$A(1, 1) = \int_{A_1} \Phi_1^2 ds + \int_{A_2} \Phi_1^2 ds + \int_{A_3} \Phi_1^3 ds.$$

Soit $\gamma : t \rightarrow a + t(b - a)$ un paramétrage de l'arête A_2 avec $a = (0, 1)$ et $b = (1, 0)$, on a alors

$$\int_{A_2} \Phi_1^2 ds = \int_0^1 \Phi_1^2(\gamma(t)) \|\gamma'(t)\|_2 dt = \sqrt{2} \int_0^1 \Phi_1^2(\gamma(t)) dt.$$

Comme $\Phi_1(\gamma(t))$ est un polynôme de degré 1 qui vaut 1 en 0 et s'annule en 1 on en déduit que $\Phi_1(\gamma(t)) = 1 - t$ et que

$$\int_{A_2} \Phi_1^2 ds = \sqrt{2} \int_0^1 (1 - t)^2 dt = \sqrt{2}/3.$$

On en déduit avec le calcul effectué dans l'exemple précédent pour $A(1, 1)$ que :

$$A(1, 1) = 2/3 + \sqrt{2}/3.$$

Pour les valeurs de $A(1, 2)$, $A(2, 1)$, $A(2, 2)$, la fonction de base Φ_2 liée au ddl numéro 2 (de coordonnées $(0, 0)$) étant nulle sur A_2 , on conserve les valeurs de l'exemple précédent :

$$A(1, 2) = A(2, 1) = 1/6, \quad A(2, 2) = 2/3.$$

Soit Φ_5 la fonction de base P_2^c de E qui vaut 1 au sommet de coordonnées $(0, 1)$ et 0 aux autres degrés de liberté du P_2^c . On a

$$A(5, 5) = \int_{A_1} \Phi_5^2 ds + \int_{A_2} \Phi_5^2 ds + \int_{A_3} \Phi_5^2 ds = 4/15 + \sqrt{2} \int_0^1 \Phi_5^2(\gamma(t)) dt.$$

Comme $\Phi_5(\gamma(t))$ est un polynôme de degré 2 qui vaut respectivement 1, 0, 0 en $t = 1, 1/2$ et 0, on en déduit que $\Phi_5(\gamma(t)) = 2t(t - 1/2)$ et que

$$\int_{A_2} \Phi_5^2 ds = \sqrt{2} \int_0^1 2^2 t^2 (t - 1/2)^2 dt = 2\sqrt{2}/15,$$

ce qui donne :

$$A(5, 5) = 4/15 + 2\sqrt{2}/15.$$

La fonction de base Φ_{10} est la fonction de base P_2^c de E qui vaut 1 au milieu du carré et 0 aux autres degrés de liberté du P_2^c . On a en raisonnant come précédemment :

$$A(10, 10) = \int_{A_2} \Phi_{10}^2 ds = \sqrt{2} \int_0^1 4t^2(1-t)^2 = 8\sqrt{2}/15.$$

Pour $F(1)$ on a avec $\gamma(t) = (\gamma_1(t), \gamma_2(t))$:

$$\begin{aligned} F(1) &= \int_{A_1} x^2 \Phi_1 ds + \int_{A_2} x^2 \Phi_1 ds + \int_{A_3} x^2 \Phi_1 ds = 1/12 + \sqrt{2} \int_0^1 \gamma_1^2(t)(1-t) dt \\ &= 1/12 + \sqrt{2} \int_0^1 t^2(1-t) dt = 1/12 + \sqrt{2}/12, \end{aligned}$$

etc..

Exemple 2. La commande

`[Ass_Ad, Sc_Ad, A, Fc_Ad, F] = exemple4_matrice_elem_Ad ;`

reprend le problème traité par la fonction `exemple3_matrice_elem_Ad` mais avec du P_2 discontinu au lieu du P_2 continu.

Donnons un exemple de calculs de termes de A . Voici celui de $A(5, 5)$. Cette fois-ci, pour le P_2 discontinu, la fonction de base Φ_5 est la fonction de base de E qui vaut 1 au sommet de coordonnées $(0, 1)$, qui vaut 0 aux autres degrés de liberté du $P_2(K_1)$ et qui est nulle sur le triangle K_2 . On a alors avec la pondération $1/2$ pour l'intégration de chaque coté de A_2

$$\begin{aligned} A(5, 5) &= \int_{A_1} \Phi_5^2 ds + \frac{1}{2} \int_{A_2} \Phi_{5|K_1}^2 ds + \frac{1}{2} \int_{A_2} \Phi_{5|K_2}^2 ds \\ &= \int_0^1 2^2 y^2 (y - 1/2)^2 dy + \frac{\sqrt{2}}{2} \int_0^1 2^2 t^2 (t - 1/2)^2 dt + \frac{1}{2} \int_{A_2} 0 ds = 2/15 + \sqrt{2}/15. \end{aligned}$$

Soit maintenant le calcul de $A(9, 9)$ et $A(14, 14)$. La fonction de base Φ_9 (resp. Φ_{14}) est la fonction de base de E qui vaut 1 au milieu du carré et 0 aux autres degrés de liberté du $P_2(K_1)$ (resp. $P_2(K_2)$) et 0 sur le triangle K_2 (resp. K_1). On trouve alors pour ces fonctions :

$$A(9, 9) = A(14, 14) = \frac{1}{2} \int_{A_2} \Phi_9^2 ds = \frac{\sqrt{2}}{2} \int_0^1 4t^2(1-t)^2 = 4\sqrt{2}/15.$$

Algorithme :

EF/Assemblage_elementaire/matrice_elem_Ad.m

On remet ici la partie déclarative de la fonction :

```
function [Sc_Fr] = matrice_elem_Ad(donnees, i, T, Tc, Pf, X, Y,...
                                Uc, D, Mat_syst_Fr, Ass_Fr, Sc_Fr) ;
```

- Boucle sur le premier triangle et le second éventuel triangle.

%On parcourt les deux triangles eventuels

```
for at = [0,2]
```

- Numéro du triangle auquel appartient l'arête i :

```
numK = Pa(i,1 + at) ;
```

- Numéro localement de l'arête i sur le triangle :

```
numALoc = Pa(i,2 + at) ;
```

- Numéro global de l'arête i :

```
numA = i ;
```

- Numéro de frontière à laquelle appartient l'arête i :

```
numF = Pa(i,5) ;
```

- On vérifie que numK est non nul :

```
if numK>0%il y a un triangle
```

- Ci-dessous `ptc{ic}` contient les numéros de *ddl* du *ic-ième* type d'espace qui appartiennent au triangle numéro numK :

```
for ic=1 :length(D.E)
```

```
    ptc{ic} = Tc{ic}(numK, :) ;
```

```
end ;
```

- Coordonnées des 3 sommets du triangle numéro numK que l'on note pour la suite a , b et c de coordonnées (a_1, a_2) , (b_1, b_2) , (c_1, c_2) :

```
XT = X(T(numK,1 :3)) ;
```

```
YT = Y(T(numK,1 :3)) ;
```

- Valeur des composantes, associées aux *ddl*, de la fonction de l'espace EF produit E représentée par Uc (Uc représente par exemple l'itéré courant d'une méthode de point fixe) :

%Boucle sur les espaces de l'espace produit

```
for ic=1 :length(D.f)
```

```
    Uc_pt{ic} = Uc{ic}(ptc{D.f(ic)}) ;
```

```
end ;
```

- Le poids est mis sur l'arête courante :

```
c = [0,0,0] ;
```

```
c(numALoc) = 1 ;
```

• Calculs de coefficients pour réaliser l'assemblage sur le triangle numéro i . Ces coefficients sont :

- Coordonnées respectives des 3 sommets du triangle : $(a_1, a_2) = (a_1, a_2)$, $(b_1, b_2) = (b_1, b_2)$, $(c_1, c_2) = (c_1, c_2)$,

- Valeur du Jacobien : $de = (a_1, b_1, c_1)(b_2 - c_2, -a_2 + c_2, -b_2 + a_2)'$,

- Coordonnées cartésiennes des points de la formule de quadrature : **Ac** et **Bc**,

- Coefficients associés au points de la formule de quadrature : **coeff**,

- Valeurs au points de la formule de quadrature des coordonnées barycentriques relatives à a, b, c données par **L1, L2, L3** :

```
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
```

```
assemble_coeff_A(XT,YT,c,D) ;
```

Les points a) b) c) portent sur le calcul des 3 types de termes parmi les 4 types de termes que calcule la fonction **pre_assemble_A** : (le 4-ième type de terme étant la matrice transposée du cas c) :

a) Calcul par type d'espace de numéro **D.E{ic}** des coefficients d'intégration sous forme dupliquée à la dimension de l'espace :

• Boucle sur les types d'espaces utilisés

```
for ic=1 :length(D.E)
```

```
    I{ic} = pre_assemble_A([], D.E(ic), [], a1, b1, c1, ...
```

```
        a2, b2, c2, de, Ac, Bc, coeff, 1, D, L1, L2, L3) ;
```

```
end ;
```

b) Contribution dans l'assemblage de l'équation, du poids de la fonction **Uc**, qui peut être par exemple l'itéré courant dans une méthode de point fixe :

• Boucle sur les espaces de l'espace produit :

```
%boucle sur les espaces de l'espace produit
```

```
for ic=1 :length(D.f)
```

```
    %boucle sur les types de derivation
```

```
    for jc=1 :length(D.der)
```

```
        uc{ic,jc} = pre_assemble_A(Uc_pt{ic},D.E(D.f(ic)), ...
```

```
            D.der(jc), a1, b1, c1, a2, b2, c2, de, Ac, Bc, ...
```

```
            coeff, 2, D, L1, L2, L3) ;
```

```
    end ;
```

```
end ;
```

c) Valeurs des fonctions de base et de leur dérivées pour chaque type d'espace à partir de la variable `Ass_Fr` qui, dans le code, a été calculée au préalable par `pre_assemble_A` :

- Boucle sur les espaces de l'espace produit :

```
%boucle sur les espaces de l'espace produit
for ic=1 :length(D.f)
    %boucle sur les types de derivation
    for jc=1 :length(D.der)%drivation
        %Ajout contribution premier ou second triangle
        sc{ic,jc} = Ass_Ad{i,D.f(ic),jc, 1 + at/2};
        tc{ic,jc} = sc{ic,jc}' ;
    end ;
end ;
```

Les points a) b) c) suivants portent sur l'assemblage de l'équation.

- a) Initialisation des matrices de type cellule décrivant les équations

```
[Mc, Nc] = feval(Mat_syst_Ad, donnees, numF, uc, a1, b1, ...
                c1, a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

- b) Assemblage de la première page

```
L2c = assemble_equ(D, I, Mc( :, :,1), Nc( :, :,1), sc, tc) ;
```

- c) Assemblage des autres pages.

```
for k=2 :size(Mc,3)
    L2cp = assemble_equ(D, I, Mc( :, :,k), Nc( :, :,k), sc, tc) ;
    for i1=1 :length(L2c( :)),
        L2c{i1} = L2c{i1} + L2cp{i1} ;
    end ;
end ;
```

- Pondération : 1 sur la frontière et 1/2 pour une arête intérieure.

```
mu = 1/2 ;
if Pa(i,5)>0
    mu = 1 ;%sur la frontiere
end
```

- Mise dans la matrice du système :

```
%Boucle sur les espaces de l'espace produit :
for ic=1 :length(D.f)
```

```

    %Boucle sur les espaces de l'espace produit :
    for jc=1 :length(D.f)
        Sc_Ad{ic,jc}(ptc{D.f(ic)},ptc{D.f(jc)}) = ...
            Sc_Ad{ic,jc}(ptc{D.f(ic)},ptc{D.f(jc)}) + mu*L2c{ic,jc};
    end
end ;
• Fin du si : les opérations ne sont effectuées que si numK est non nul.
end ;%if numK>0
• Fin de la boucle sur le premier triangle et le second éventuel triangle.
end ;%for at

```

Fonction :

```
function [Fc_Dom_scm] = matrice_elem_T_scm(donnees, i, T, Tc, ...
      X, Y, Uc, D, Mat_syst_fscm_Dom, Ass_Dom, Fc_Dom_scm) ;
```

Calcul de matrices élémentaires pour l'assemblage sur les triangles de la forme linéaire associée à l'EDP (second membre). C'est la partie en b_Ω de la forme linéaire $\mathcal{B}$ dans (1.10).

Pour un triangle donné (désigné ci-dessus par son numéro i), cette fonction calcule la matrice élémentaire relative à la forme linéaire associée au second membre du système d'EDP (hors termes d'arêtes). La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP (désignée ci-dessus par `Fc_Dom_scm`, matrice de type CELL), qui est une représentation par blocs de vecteur.

La commande

```
[Fc_Dom_scm] = matrice_elem_T_scm(donnees, i, T, Tc, ...
      X, Y, Uc, D, Mat_syst_fscm_Dom, Ass_Dom, Fc_Dom_scm) ;
```

permet de calculer le second membre élémentaire, par blocs, du système relativement au triangle numéro i , que l'on notera aussi i , de sommets $T(i, 1 : 3)$ et l'ajoute à la matrice globale du système `Fc_Dom_scm`, matrice de type CELL, représentation par blocs de vecteurs.

La variable `Mat_syst_fscm_Dom` ci-dessus un handle de fonction (ou un nom de fonction) décrivant le second membre du système d'équations aux dérivées partielles sur le domaine (hors termes d'arêtes). Les autres variables d'entrées sont les mêmes que celles de la fonction `matrice_elem_T`.

Variables d'entrée :

Les variables sont identiques à celles de la fonction `matrice_elem_T` excepté la variable `Fc_Dom_scm` qui représente une contribution au vecteur second membre de l'EDP alors que la variable de sortie `Sc_Dom` de la fonction `matrice_elem_T` est une contribution à la matrice du système. D'autre part, la fonction désignée par `Mat_syst_Dom_fscm`, qui est une fonction définie par l'utilisateur, a une syntaxe différente de la fonction appelée dans `matrice_elem_T` par la variable `Mat_syst_Dom`, avec ici l'ajout d'une variable de sortie supplémentaire.

La syntaxe de la fonction désignée par `Mat_syst_fscm_Dom`, que nous notons ci-dessous `scm_Dom`, est la suivante :

Syntaxe de la fonction appelée par `matrice_elem_T_scm`.

```
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, ...
      a2, b2, c2, de, Ac, Bc, numK) ;
```

Les variables sont identiques à celles de la fonction appelée par `matrice_elem_T`, avec en plus une variable de sortie `Fc` pour le second membre du système. Nous précisons alors les variables de sortie ci-dessous.

`Mc`, `Nc`, `Fc` : CELL ARRAY. Les variables de sortie `Mc` et `Nc` sont les variables de type `cell` décrivant l'EDP sur le domaine et `Fc` celle qui décrit le second membre de l'EDP. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de `matrice_elem_T_scm` :

`Fc_Dom_scm` : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Représente la matrice par blocs de l'assemblage de la partie en Ω de la forme bilinéaire $\mathcal{B}$ (terme en b_Ω) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Dom` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les triangles avec utilisation de la fonction `matrice_elem_T_scm`. Le terme `Fc_Dom_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`.

Variables de sortie :

`Fc_Dom_scm` : CELL ARRAY. Vecteur second membre du système augmentée de la matrice élémentaire calculée par `matrice_elem_T_scm`.

Exemple 1 :

L'exemple qui suit reprend l'exemple `exemple1_matrice_elem_T` donné dans la présentation de la fonction `matrice_elem_T`. Le domaine Ω est le carré maillé par le triangle K_1 de sommets $(0,1)$, $(0,0)$ et $(1,0)$ et le triangle K_2 de sommets $(0,1)$, $(1,0)$ et $(1,1)$ (cf. Figure 4.3-10) et E est l'espace EF produit donné par $E = P_1^c \times P_2^c \times P_1^c$ sur ce maillage.

A l'assemblage de la forme bilinéaire du système est ajouté à titre d'exemple le calcul du second membre f du système avec

$$f = (f_1(x, y), f_2(x, y), f_3(x, y)) = (x^2, x^2 + y^2, x).$$

Le système obtenu est résolu dans l'exemple, sa solution, notée $u = (u_1, u_2, u_3)$, est donc la projection orthogonale $[L^2(\Omega)]^3$ de f sur l'espace $E = P_1^c \times P_2^c \times P_1^c$. Donc u_1 est la projection $L^2(\Omega)$ de f_1 sur l'espace P_1^c et pour les deux autres composantes on a $u_2 = f_2$ et $u_3 = f_3$ par unicité de la projection orthogonale car (f_2, f_3) appartient elle-même à l'espace de projection $E_2 \times E_3$.

L'exemple est donné dans le fichier `exemple1_matrice_elem_T_scm` du répertoire EF/exemples/Assemblage_elementaire, et se met en œuvre par la commande

```
[Uc, Xc, Yc] = exemple1_matrice_elem_T_scm ;
```


L'erreur entre la fonction x^2 et son projeté sur le P_1^c est représentée à la figure 4.3-14 suivante.

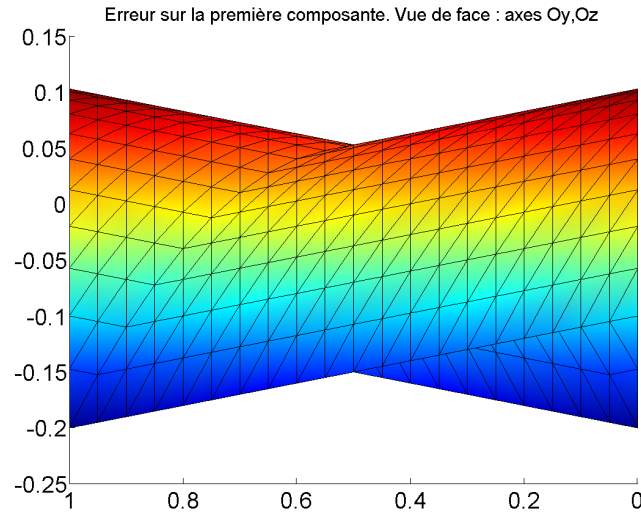


Figure 4.3-14. Graphe de l'écart entre la fonction x^2 et sa projection orthogonale sur le P_1 continu. Le plan Oy, Oz est vu de face. Le domaine est le carré unité maillé par par deux triangles.

Algorithme :

EF/Assemblage_elementaire/matrice_elem_T_scm.m

On remet ici la partie déclarative de la fonction :

```
function [Fc_Dom_scm] = matrice_elem_T_scm(donnees, i, T, Tc, ...
    X, Y, Uc, D, Mat_syst_fscm_Dom, Ass_Dom, Fc_Dom_scm) ;
```

l'algorithme est identique à celui de la fonction `matrice_elem_T`, excepté la dernière partie concernant l'assemblage proprement dit. Cette partie est décrite ci-dessous.

a) b) c) : Assemblage de l'équation.

- a) Initialisation des matrices de type cellule décrivant les équations :

```
[Mc, Nc, Fc] =feval(Mat_syst_fscm_Dom, donnees, uc, a1, ...
    b1, c1, a2, b2, c2, de, Ac, Bc, i) ;
```

- b) La structure de `Fc` est donnée au Tableau 2.2.4-1 du Paragraphe 2.2.4 (deuxième partie). La variable `Fc` est un tableau $1 \times 3 \times n_D$ de cellules où n_D est le nombre de pages (ici $D = \Omega$ le domaine d'intégration). Pour un numéro de page k donné `Fc{1,i,k}` est un vecteur colonne de m_D^k cellules, chaque cellule

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_T_scm

contient un vecteur ligne de type `double`. Par exemple pour $i = 1, 2, 3$, $\text{Fc}\{1, i, k\}$ est de la forme

$$\text{Fc}\{1, i, k\} = \begin{bmatrix} 1 \times r \text{ double} \\ 1 \times r \text{ double} \\ \dots \\ 1 \times r \text{ double} \end{bmatrix}$$

avec m_D^k lignes, où r est en général le nombre de points de la formule de quadrature. Ainsi la variable $[\text{Fc}\{1, 1, k\}, \text{Fc}\{1, 2, k\}, \text{Fc}\{1, 3, k\}]$ est la variable de type `CELL` de taille $m_D^k \times 3$ de la forme

$$[\text{Fc}\{1, 1, k\}, \text{Fc}\{1, 2, k\}, \text{Fc}\{1, 3, k\}] = \begin{bmatrix} 1 \times r \text{ double} & 1 \times r \text{ double} & 1 \times r \text{ double} \\ 1 \times r \text{ double} & 1 \times r \text{ double} & 1 \times r \text{ double} \\ \dots & \dots & \dots \\ 1 \times r \text{ double} & 1 \times r \text{ double} & 1 \times r \text{ double} \end{bmatrix}.$$

On initialise alors Z ci-dessous en transposant chacun des vecteurs ci-dessus par :

$$Z = \begin{bmatrix} r \times 1 \text{ double} & r \times 1 \text{ double} & r \times 1 \text{ double} \\ r \times 1 \text{ double} & r \times 1 \text{ double} & r \times 1 \text{ double} \\ \dots & \dots & \dots \\ r \times 1 \text{ double} & r \times 1 \text{ double} & r \times 1 \text{ double} \end{bmatrix}$$

de taille m_D^k lignes et 3 colonnes.

On transpose ainsi le contenu des cellules $\text{Fc}\{1, i, k\}$ de chaque page de Fc (vecteurs lignes transformés en vecteurs colonnes), Et chaque page est associée à 3 seconds membres mis dans les colonnes de Z . Ci-dessous dans l'algorithme de la fonction, le cas le première page :

```
Z = Fc( :, :, 1 );
%test sur le contenu des pages de Fc : les Fc(1,i,k) doivent
%  etre des vecteurs colonnes
if any ( [size(Z{1,1},2)~=1, size(Z{1,2},2)~=1, size(Z{1,3},2)~=1] )
    error(['Fc(1,1,1), Fc(1,2,1), Fc(1,3,1) doivent etre des ' ...
          ' vecteurs colonnes de cellules'])
end ;
Z = [Z{1,1}, Z{1,2}, Z{1,3}] ;
[m1, n1] = size(Z) ;
for i1=1 :m1 ;
    for j1=1 :n1
        Z{i1, j1} = Z{i1, j1}' ;
```

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_T_scm

```

    end ;
end ;
• b) Calcul de la matrice élémentaire associée à la première page
L2c = assemble_equ(D,I, Mc( :, :,1), Nc( :, :,1), Z, tc) ;
La{1} = L2c ;
• c) Traitement des autres pages suivant le même principe avec une boucle sur  $k$ 
le numéro de page,  $Z = Fc(:, :, k)$  étant la page  $k$  :
for k=2 :size(Mc,3)%
    Z = Fc( :, :,k) ;%tableau de 3 cellules valeur de f, fx, fy
                                %sur cette arete
    Z = [Z{1,1},Z{1,2},Z{1,3}] ;
    [m1,n1] = size(Z) ;
    for i1=1 :m1 ;
        for j1=1 :n1
            Z{i1,j1} = Z{i1,j1}' ;
        end ;
    end ;
    L2cp = assemble_equ(D, I, Mc( :, :,k), Nc( :, :,k), Z, tc) ;
    La{k} = L2cp ;
end ;

```

Le nombre de lignes m_D^k que contient $Fc\{1, i, k\}$, $1 \leq i \leq 3$, peut varier avec le numéro de page k . Ceci est pris en compte dans l'algorithme.

- Mise dans la matrice du système

```

for k=1 :size(Mc,3)
    L2c = La{k} ;
    for ic=1 :size(L2c,1) ;%boucle sur les espaces de l'espace
                                %produit
        for jc=1 :size(L2c,2) ;%boucle sur le nombre de seconds
                                %membres
            Fc_Dom_scm{ic,1}(ptc{D.f(ic)},1 ) = ...
                Fc_Dom_scm{ic,1}(ptc{D.f(ic)},1) + L2c{ic,jc} ;
        end ;
    end ;
end ;

```

Fonction :

```
function [Fc_Fr_scm] = matrice_elem_A_scm(donnees, i, T, Tc, ...
    Pf, X, Y, Uc, D, Mat_syst_fscm_Fr, Ass_Fr, Fc_Fr_scm) ;
```

Calcul de matrices élémentaires pour l'assemblage de la forme linéaire sur la frontière associées à l'EDP. C'est la partie en b_Γ de la forme linéaire $\mathcal{B}$ dans (1.10).

Cette fonction est analogue à la fonction `matrice_elem_T_scm`, avec une variable d'entrée supplémentaire, la variable `Pf` (cf. Tableau 2.2.3-1b) qui décrit les arêtes de la frontière suivant une numérotation, localement sur la frontière, des arêtes. La variable `i`, que l'on notera aussi i , indique dans cette numérotation le numéro d'arête sur laquelle on intègre.

Pour une arête frontière donnée numéro i , située sur le triangle $Pf(i,1)$, la commande

```
[Fc_Fr_scm] = matrice_elem_A_scm(donnees, i, T, Tc, ...
    Pf, X, Y, Uc, D, Mat_syst_fscm_Fr, Ass_Fr, Fc_Fr_scm) ;
```

permet de calculer la matrice élémentaire relative à la forme bilinéaire associée au système d'EDP et aux termes d'arêtes frontières. La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP et aux arêtes frontières (désignée ci-dessus par `Fc_Fr_scm`, vecteur de type `CELL`).

La variable `Mat_syst_fscm_Fr` ci-dessus un handle de fonction (ou un nom de fonction) décrivant le second membre du système d'équations aux dérivées partielles sur les arêtes frontières (hormis les termes traités par la fonction `matrice_elem_Ad_scm`). Les autres variables d'entrées sont les mêmes que celles de la fonction `matrice_elem_A`. Précisons ces variables.

Variables d'entrée :

Les variables sont identiques à celles de la fonction `matrice_elem_A` excepté la variable `Fc_Fr_scm` qui représente une contribution au vecteur second membre de l'EDP alors que la variable de sortie `Sc_Fr` de la fonction `matrice_elem_A` est une contribution à la matrice du système. D'autre part, la fonction désignée par `Mat_syst_Fr_fscm`, qui est une fonction définie par l'utilisateur, a une syntaxe différente de la fonction appelée dans `matrice_elem_A` par la variable `Mat_syst_Fr`, avec ici l'ajout d'une variable de sortie supplémentaire.

La syntaxe de la fonction désignée par `Mat_syst_fscm_Fr`, que nous notons ci-dessous `scm_Fr`, est la suivante :

Syntaxe de la fonction appelée par `matrice_elem_A_scm`.

```
function [Mc, Nc, Fc] = scm_Fr(donnees, numF, uc, a1, b1, c1, ...
    a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_A_scm

Les variables sont identiques à celles de la fonction appelée par `matrice_elem_A`, avec en plus une variable de sortie `Fc` pour le second membre du système. Nous précisons alors les variables de sortie ci-dessous.

`Mc`, `Nc`, `Fc` : CELL ARRAY. Les variables de sortie `Mc` et `Nc` sont les variables de type `cell` décrivant l'EDP sur le domaine et `Fc` celle qui décrit le second membre de l'EDP. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de `matrice_elem_A_scm` :

`Fc_Fr_scm` : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Analogue de `Fc_Dom_scm` relativement à la partie sur Γ de la forme bilinéaire $\mathcal{B}$. Représente la matrice par blocs de l'assemblage de la partie en Γ de la forme bilinéaire $\mathcal{B}$ (terme en b_Γ) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Fr` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les arêtes frontières avec utilisation de la fonction `matrice_elem_A_scm`. Le terme `Fc_Fr_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`.

Variables de sortie :

`Fc_Fr_scm` : CELL ARRAY. Vecteur second membre du système augmentée de la matrice élémentaire calculée par `matrice_elem_A_scm`.

Exemple : Voir à la description de la fonction `matrice_elem_A`, l'exemple donné par la commande :

```
[Ass_Fr, Sc_Fr, A, Fc_Fr, F] = exemple3_matrice_elem_A ;
```

où la fonction `matrice_elem_A_scm` est utilisée pour le calcul du second membre du système. Le calcul des termes $F(1)$ à $F(6)$ est détaillé dans le descriptif de l'exemple.

Algorithme.

L'algorithme suit le même principe que celui de la fonction `matrice_elem_A`, la partie assemblage du système est remplacée par une partie traitement du second membre, comme effectué dans la fonction `matrice_elem_T_scm` (voir la description de l'algorithme de ces deux fonctions).

Fonction :

```
function [Fc_Ad_scm] = matrice_elem_Ad_scm(donnees, i, T, Tc, ...
      Pa, X, Y, Uc, D, Mat_syst_fscm_Ad, Ass_Ad, Fc_Ad_scm) ;
```

En développement, fonctionne. Calcul de matrices élémentaires pour l'assemblage de la forme linéaire sur les arêtes du maillage associée à l'EDP. C'est la partie en b_{Σ} de la forme linéaire $\mathcal{B}$ dans (1.10).

Cette fonction est analogue à la fonction `matrice_elem_T_scm`, avec une variable d'entrée supplémentaire, la variable `Pa` (cf. Tableau 2.2.3-1b) qui décrit les arêtes du maillage dans la numérotation globale des arêtes du domaine. La variable `i`, que l'on notera aussi i , indique dans cette numérotation le numéro d'arête sur laquelle on intègre.

Pour le domaine d'intégration qui assemble la matrice du système : l'intégration se fait sur la frontière Γ et sur Σ la réunion des arêtes intérieures, avec

$$\int_{\Sigma} f = \frac{1}{2} \int_{\Sigma_1} f + \frac{1}{2} \int_{\Sigma_2} f,$$

où on note Σ_p , $p = 1, 2$ la réunion des arêtes intérieures, avec la convention que sur une arête i de Σ_1 (resp. Σ_2) on intègre la restriction de f au triangle de numéro $\text{Pa}(i, 1)$ (resp. $\text{Pa}(i, 3)$). On pourra aussi se reporter à la description de la fonction `matrice_elem_Ad` et aux calculs effectués dans les exemples.

Pour une arête numéro i , située sur le triangle de numéro $\text{Pa}(i, 1)$ ou bien $\text{Pa}(i, 3)$, la commande

```
[Fc_Ad_scm] = matrice_elem_Ad_scm(donnees, i, T, Tc, ...
      Pa, X, Y, Uc, D, Mat_syst_fscm_Ad, Ass_Ad, Fc_Ad_scm) ;
```

permet de calculer la matrice élémentaire relative à la forme bilinéaire associée au système d'EDP et aux termes d'arêtes frontières. La contribution de cette matrice élémentaire est ajoutée à la matrice EF globale associée à l'EDP et aux arêtes du domaine (désignée ci-dessus par `Fc_Ad_scm`, vecteur de type `CELL`).

On note que comme dans la fonction `matrice_elem_A_scm` vue précédemment, il y a ici aussi une prise en compte d'arêtes sur la frontières.

La variable `Mat_syst_fscm_Ad` ci-dessus est un handle de fonction (ou un nom de fonction) décrivant le second membre du système d'équations aux dérivées partielles sur les arêtes (hormis les termes traités par la fonction `matrice_elem_A_scm`). Les autres variables d'entrée sont les mêmes que celles de la fonction `matrice_elem_Ad`. Précisons ces variables.

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_Ad_scm

Variables d'entrée :

Les variables sont identiques à celles de la fonction `matrice_elem_Ad` excepté la variable `Fc_Ad_scm` qui représente une contribution au vecteur second membre de l'EDP alors que la variable de sortie `Sc_Ad` de la fonction `matrice_elem_Ad` est une contribution à la matrice du système. D'autre part, la fonction désignée par `Mat_syst_Ad_fscm`, qui est une fonction définie par l'utilisateur, a une syntaxe différente de la fonction appelée dans `matrice_elem_Ad` par la variable `Mat_syst_Ad`, avec ici l'ajout d'une variable de sortie supplémentaire.

La syntaxe de la fonction désignée par `Mat_syst_fscm_Ad`, que nous notons ci-dessous `scm_Ad`, est la suivante :

Syntaxe de la fonction appelée par `matrice_elem_Ad_scm`.

```
function [Mc, Nc, Fc] = scm_Ad(donnees, numF, uc, a1, b1, c1, ...
                             a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

Les variables sont identiques à celles de la fonction appelée par `matrice_elem_Ad`, avec en plus une variable de sortie `Fc` pour le second membre du système. Nous précisons alors les variables de sortie ci-dessous.

`Mc, Nc, Fc` : CELL ARRAY. Les variables de sortie `Mc` et `Nc` sont les variables de type `cell` décrivant l'EDP sur le domaine et `Fc` celle qui décrit le second membre de l'EDP. Ces variables sont décrites au Tableau 2.2.4-1, seconde partie, du Paragraphe 2.2.4.

Suite de la description des variables d'entrée de `matrice_elem_Ad_scm` :

`Fc_Ad_scm` : CELL ARRAY. Matrice de taille $n \times 1$ avec n nombre de termes de l'espace EF produit E . Analogue de `Fc_Dom_scm` relativement à la partie sur $\bar{\Sigma}$ (réunion de toutes les arêtes du maillage) de la forme bilinéaire $\mathcal{B}$. Représente la matrice par blocs de l'assemblage de la partie en $\bar{\Sigma}$ de la forme bilinéaire $\mathcal{B}$ (terme en $b_{\bar{\Sigma}}$) de (1.10), Paragraphe 1.2. La forme linéaire est la fonction désignée par le champ `donnees.fscm_Ad` (cf. Tableau 2.2.1-1b du Paragraphe 2.2.1). L'assemblage est réalisé par boucle sur les arêtes du maillage avec utilisation de la fonction `matrice_elem_Ad_scm`. Le terme `Fc_Ad_scm{i}` est le vecteur colonne (SPARSE ARRAY) de taille $\dim(E_i)$ correspondant à l'assemblage sur E_i , i -ème espace de l'espace EF produit E de numéro `donnees.D.E(donnees.D.f(i))`. En développement, fonctionne.

Variables de sortie :

`Fc_Ad_scm` : CELL ARRAY. Vecteur second membre du système augmentée de la matrice élémentaire calculée par `matrice_elem_Ad_scm`.

Suite de la description des variables d'entrée de `matrice_elem_A_scm` :

Exemple :

Voir l'exemple 1 décrit dans la fonction `matrice_elem_Ad` avec la commande

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_scm

```
[Ass_Ad, Sc_Ad, A, Fc_Ad, F] = exemple3_matrice_elem_Ad ;
```

où la fonction `matrice_elem_Ad_scm` est utilisée pour le calcul du second membre du système. Le calcul de $F(1)$ est détaillé dans l'exemple.

Voir aussi la commande

```
[Ass_Ad, Sc_Ad, A, Fc_Ad, F] = exemple4_matrice_elem_Ad ;
```

de l'exemple 2 de la description de `matrice_elem_Ad`.

Algorithme.

L'algorithme suit le même principe que celui de la fonction `matrice_elem_Ad`, la partie assemblage du système est remplacée par une partie traitement du second membre, comme effectué dans la fonction `matrice_elem_T_scm` (voir la description de l'algorithme de ces deux fonctions).

Fonction :

```
function [Ass_Dom, SL_Dom, f_Dom]=matrice_elem_prepa_T(donnees, ...
    i, T, Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;
```

Calcul de valeurs de fonctions de base d'un triangle, de matrices d'assemblage type norme d'énergie (contribution du triangle) et graphe des matrices d'assemblage (contribution du triangle). Ci-dessous *i* désignera le numéro du triangle courant et *i* une variable générique quelconque.

On note F_k l'espace d'EF dont le numéro est donné par $D.E(k)$ et $\text{length}(D.E)$ est le nombre de type d'espace d'EF F_k utilisés.

La commande

```
[Ass_Dom, SL_Dom, f_Dom] = matrice_elem_prepa_T(donnees, ...
    i, T, Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;
```

- calcule les valeurs des fonctions de base et de leurs dérivées aux points de la formule de quadrature de numéro $D.intT$ sur le triangle K de numéro *i*, de sommets dont les numéros sont donnés par $T(i, 1 : 3)$ et met le résultat dans $Ass\_Dom\{i, j, k + 1\}$ avec $1 \leq j \leq \text{length}(D.E)$, et $0 \leq k \leq 2$ désignant les ordres de dérivations ($\partial_0, \partial_x, \partial_y$).

- calcule la contribution du triangle K à la matrice d'assemblage des formes bilinéaires

$$a_{ij}(f, g) = \int_{\Omega} (f_{,k}, g_{,m})$$

définies sur $F_j \times F_i$ avec $0 \leq k, m \leq 2$ les ordres de dérivation. La contribution du triangle K est ajouté aux termes de la matrice $SL\_Dom\{i, j, k + 1, m + 1\}$ correspondant aux *ddl* globaux concernés par la matrice d'assemblage elementaire.

- calcule la contribution du triangle K au graphe des matrices d'assemblage pour des formes bilinéaires sur $F_j \times F_i$ et met le résultat dans les termes de $f\_Dom\{i, j\}$ correspondant aux *ddl* concernés. Lorsque l'on a parcouru tous les triangles avec la fonction `matrice_elem_prepa_T` on a alors $f\_Dom\{i, j\}(p, q) = 1$ si l'intersection des supports de la *p*-ième fonction de base de F_i et de la *q*-ième fonction de base F_j est de mesure non nulle et $f\_Dom\{i, j\}(p, q) = 0$ sinon. Dans le code EF cette variable sert uniquement à estimer la mémoire à allouer aux matrices d'assemblage et ce pour un gain de temps.

La variable `donnees` est la variable de type `struct` qui représente les données du problème, la variable *i* est le numéro du triangle courant, *T* est le tableau qui décrit le maillage, *Tc* représente les degrés de liberté des différents types d'espaces, *X* et *Y* les coordonnées en *x* et en *y* des sommets du maillage, *D* est une variable de type `struct` décrivant les espaces d'EF et les formules d'intégration, utilisée pour ses champs *D.E*, *D.der* et *D.intT*.

La variable *Ass_Dom* contient les valeurs des fonctions de base ainsi que de leurs dérivées aux points de la formule de quadrature sur le triangle courant.

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_prepa_T

La variable `SL_Dom` est un tableau de type tableau de cellules qui représentent chacune la matrice globale d'assemblage et `f_Dom` le graphe des termes non nul dans les matrices d'assemblage.

Variables d'entrée :

`donnees` : `STRUCT ARRAY`. Variable décrivant les données du problème EF. Facultative ici. Peut être mise à `[]`.

`i` : `DOUBLE ARRAY`. Numéro du triangle courant.

`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`Tc` : `CELL ARRAY`. Vecteur ligne de cellules. Les cellules sont des matrices de type `DOUBLE ARRAY`. La matrice `Tc{i}` décrit triangle par triangle les *ddl* du *i*-ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

`X,Y` : `DOUBLE ARRAY`. Coordonnées des sommets du maillage, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

`D` : `STRUCT ARRAY`. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ `D` du Tableau 2.2.1-1c, Paragraphe 2.2.1. En pratique `D = donnees.D`.

Les champs de `D` utilisés sont les suivants :

`D.E` : `DOUBLE ARRAY`. Vecteur des numéros des types d'espaces d'EF utilisés.

`D.der` : `DOUBLE ARRAY`. Choix des dérivations. Toujours fixé pour ce code à `[0, 1, 2]`. Cette variable doit être bien fixée à `[0, 1, 2]` sinon, par exemple, si `D.der = []` et `Ass_Dom = []` en entrée alors la variable de sortie `Ass_Dom` sera vide en sortie.

`D.intT` : `DOUBLE ARRAY`. Entier. Choix de la formule de quadrature utilisée sur les triangles.

La commande ci-dessus initialise le terme `Ass_Dom{i, j, k}` de la matrice `Ass_Dom` ci-dessous.

`Ass_Dom` : `CELL ARRAY`. Tableau de cellules $N \times N_E \times 3$ où N est le nombre de triangles et $N_E = \text{length}(\text{donnees.D.E})$ est le nombre de type d'espace utilisés (*cf.* le début du Paragraphe 2.2). Le terme `Ass_Dom{i, j, k}` est le tableau des valeurs, aux points de la formule de quadrature sur le triangle K numéro i , de la dérivée d'ordre $k-1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par `donnees.D.E(j)`. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro `donnees.D.intT` utilisée sur les triangles et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ `Ass_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

En entrée de `matrice_elem_prepa_T` on peut poser `Ass_Dom = []`, par exemple lorsque que l'on débute une boucle sur l'ensemble des triangles pour le calcul

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_T

complet par la fonction `matrice_elem_prepa_T` de la variable `Ass_Dom` telle que décrite ci-dessus.

La commande ci-dessus calcule la contribution du triangle numéro `i` à la matrice `SL_Dom` décrite ci-dessous.

`SL_Dom` : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par `donnees.D.E(k)`. Le terme `SL_Dom{i,j,k,m}` est une matrice de type SPARSE ARRAY de dimension $\dim(F_i) \times \dim(F_j)$. La matrice `SL_Dom{i,j,k,m}` est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par

$$\int_{\Omega} v_{,k-1} w_{,m-1}.$$

Le terme (p, q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$, où $\{u^k\}_k$ désigne la base de l'espace d'EF F_k . Les bases sont décrites au Paragraphe 1.4 et au Paragraphe 2.3.2, sur un maillage exemple. Cette variable représente le champ `SL_Dom` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

La commande ci-dessus calcule la contribution du triangle numéro `i` à la matrice `f_Dom` décrite ci-dessous.

`f_Dom` : CELL ARRAY. Matrice de taille $N_E \times N_E$ avec $N_E = \text{length}(\text{donnees.D.E})$. Soit F_k l'espace d'EF dont le numéro est donné par `donnees.D.E(k)`, la variable `f_Dom{i,j}` est de type SPARSE ARRAY et de taille $\dim(F_i) \times \dim(F_j)$ et le terme `f_Dom{i,j}(p,q)` vaut 1 si l'intersection des supports de la p -ième fonction de base de F_i et de la q -ième fonction de base F_j est de mesure non nulle et `f_Dom{i,j}(p,q) = 0` sinon. Cette variable représente aussi le graphe des termes *a priori* non nuls des matrices d'assemblage pour des formes bilinéaires définies sur $F_j \times F_i$, i.e. le graphe de `SL_Dom{i,j,k,m}` ci-dessus : `f_Dom{i,j}` vaut 1 quand `SL_Dom{i,j,1,1}` est *a priori* non nul (on regarde uniquement l'intersection des supports, mais pas les valeurs effectives des intégrales).

Variables de sortie :

`Ass_Dom` : CELL ARRAY. Matrice ci-dessus pour laquelle on a pris en compte la contribution du triangle numéro `i`.

`SL_Dom` : CELL ARRAY. Matrice ci-dessus pour laquelle on a pris en compte, par ajout de matrices élémentaires, de la contribution du triangle numéro `i`.

`f_Dom` : CELL ARRAY. Matrice ci-dessus pour laquelle on a pris en compte, par ajout, la contribution du triangle numéro `i`.

Exemple 1 : L'exemple suivant est tiré du `help` de la fonction `matrice_elem_prepa_T`. Cet exemple calcule les valeurs des fonctions de base et de leurs dérivées à l'ordre 1 aux points de formules de quadrature d'un maillage composé d'un seul triangle K . Les coordonnées des sommets du triangle (et du maillage)

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_prepa_T

sont dans l'ordre de numérotation $(0,0)$, $(1,0)$, $(0,1)$. Les espaces sont le $P_1^c(K)$ et $P_2^c(K)$.

EXEMPLE 1

```
%Sur un triangle :
clear
%Maillage par un triangle
T = [1,2,3] ;
X = [0,1,0]
Y = [0,0,1] ;
%DDL du P1 et du P2
Tc{1} = T ;
Tc{2} = [1,2,3,4,5,6]
%Derivations
D.der = [0,1,2]
%Types d'espaces utilises
D.E = [1,2] ;
%Formules de quadrature
D.intT = 4 ;
donnees = [] ;
%Dimension globale (locale aussi ici)
Ed = [3,6]
%Initialisations
for d1=1 :3;%Nombre de derivations
    for d2=1 :3;%Nombre de derivations
        for i=1 :length(D.E)%Nombre de types d'espaces
            for j=1 :length(D.E)
                SL_Dom{i,j,d1,d2} = sparse(Ed(i),Ed(j)) ;
            end ;
        end ;
    end ;
end ;
for i=1 :length(D.E)
    for j=1 :length(D.E)
        f_Dom{i,j} = sparse(Ed(i),Ed(j)) ;
```

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_T

```

end ;
end ;
Ass_Dom = [] ;
[Ass_Dom, SL_Dom, f_Dom] = matrice_elem_prepa_T(donnees, ...
1, T, Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;

```

Décrivons les résultats obtenus. Soient $\{\Phi_i\}_{1 \leq i \leq 3}$ les fonctions de base de $E_1 = P_1^c$ et $\{\Psi_i\}_{1 \leq i \leq 6}$ les fonctions de base de $E_2 = P_2^c$ (remarque : dans le cas d'un seul triangle, EF continu et discontinu sont identiques). Notons $m \in [\mathbb{R}^2]^{15}$ le vecteur dont les composantes sont, dans l'ordre choisi dans le code, les coordonnées des points de la formule de quadrature. Ces points sont représentés à la Figure 4.3 – 8 ci-dessus (*cf.* description de la fonction `assemble_coeff_T`), nous redonnons ci-dessous cette figure par commodité

Points de la formule de quadrature dans leur ordre de numérotation

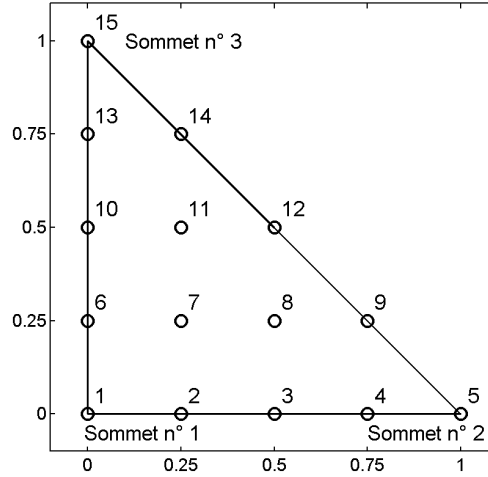


Figure 4.3-8 (figure donnée à la description de la fonction `assemble_coeff_T`).

Avec ce choix de T_c (*cf.* Paragraphe 2.3), les *ddl* de E_1 sont successivement les 3 sommets du triangle K , les *ddl* de E_2 sont successivement les 3 sommets du triangles, puis les milieux des sommets 1 et 2, des sommets 1 et 3, et des sommets 2 et 3 du triangle K .

Voici ce qu'on trouve pour les exemples suivants de fonctions de base et de dérivations :

$$\begin{aligned}
\Phi_1(m) &= \text{Ass\_Dom}\{1, 1, 1\}(:, 1) = (1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, \frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}, 0, 0)' , \\
\Phi_2(m) &= \text{Ass\_Dom}\{1, 1, 1\}(:, 2) = (0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 0, \frac{1}{4}, \frac{1}{2}, 0, \frac{1}{4}, 0)' , \\
\Phi_3(m) &= \text{Ass\_Dom}\{1, 1, 1\}(:, 3) = (0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4})' , \\
\Phi_{1,x}(m) &= \text{Ass\_Dom}\{1, 1, 2\}(:, 1) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\
\Phi_{1,y}(m) &= \text{Ass\_Dom}\{1, 1, 3\}(:, 1) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\
\Psi_1(m) &= \text{Ass\_Dom}\{1, 2, 1\}(:, 1) = (1, \frac{3}{8}, 0, -\frac{1}{8}, 0, \frac{3}{8}, 0, -\frac{1}{8}, 0, 0, -\frac{1}{8}, 0, -\frac{1}{8}, 0, 0)' , \\
&\text{etc.}
\end{aligned}$$

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_prepa_T

Ensuite, voici un exemple de calcul des matrices associées aux formes bilinéaires sur $E_1 \times E_1$ et qui sont stockées dans $\text{SL\_Dom}\{1, 1, :, :\}$ (à titre de vérification, on pourra refaire les calculs comme indiqué dans le premier exemple de la description de la fonction `matrice_elem_T`, dans le cas de deux triangles et du calcul du terme $A(1 : 4, 1 : 4)$) :

$$\int_K \Phi_j \Phi_i = \text{SL\_Dom}\{1, 1, 1, 1\}(i, j) = \begin{pmatrix} 1/12 & 1/24 & 1/24 \\ 1/24 & 1/12 & 1/24 \\ 1/24 & 1/24 & 1/12 \end{pmatrix}.$$

On vérifie que la somme des termes de la matrice est égale à la surface du triangle K *i.e.* $1/2$. Cela est dû au fait que cette somme est égale à

$$\sum_{1 \leq i, j \leq 3} \int_K \Phi_j \Phi_i = \int_K (\Phi_1 + \Phi_2 + \Phi_3)^2 = \int_K 1^2 = 1/2.$$

Calcul de matrices où interviennent des dérivées :

$$\begin{aligned} \int_K \Phi_{j,x} \Phi_{i,y} &= \text{SL\_Dom}\{1, 1, 1, 2\}(i, j) \\ &= \begin{pmatrix} -1/6 & -1/6 & -1/6 \\ 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 \end{pmatrix} = \text{SL\_Dom}\{1, 1, 2, 1\}(j, i). \end{aligned}$$

On vérifie que la somme des termes de la matrice est nulle. Cela est dû au fait que cette somme est égale à

$$\sum_{1 \leq i, j \leq 3} \int_K \Phi_{j,x} \Phi_{i,y} = \int_K (\Phi_1 + \Phi_2 + \Phi_3)_{,x} (\Phi_1 + \Phi_2 + \Phi_3)_{,y} = \int_K 1_{,x} 1_{,y} = 0.$$

Puis, exemple de calcul des matrices associées aux formes bilinéaires sur $E_2 \times E_1$ stockées dans $\text{SL\_Dom}\{1, 2, :, :\}$:

$$\begin{aligned} \int_K \Psi_j \Phi_i &= \text{SL\_Dom}\{1, 2, 1, 1\}(i, j) \\ &= \begin{pmatrix} 1/60 & -1/120 & -1/120 & 1/15 & 1/15 & 1/30 \\ -1/120 & 1/60 & -1/120 & 1/15 & 1/30 & 1/15 \\ -1/120 & -1/120 & 1/60 & 1/30 & 1/15 & 1/15 \end{pmatrix} \\ &= \text{SL\_Dom}\{2, 1, 1, 1\}(j, i) \text{ (par symétrie de la forme bilinéaire).} \end{aligned}$$

On vérifie aussi comme pour le cas du P_1 que la somme des termes de cette matrice est égale à la surface de K :

$$\begin{aligned} \sum_{1 \leq j \leq 6} \left(\sum_{1 \leq i \leq 3} \int_K \Psi_j \Phi_i \right) &= \\ \int_K (\Psi_1 + \Psi_2 + \Psi_3 + \Psi_4 + \Psi_5 + \Psi_6) (\Phi_1 + \Phi_2 + \Phi_3) &= \int_K 1^2 = 1/2. \end{aligned}$$

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_T

Exemple 1 bis : Cet exemple reprend l'exemple ci-dessus. Le tableau Tc des *ddl* ci-dessus est calculé directement par la fonction *ddl.m* qui redonne la même valeur de Tc que précédemment (le choix précédent est effectué sur le même principe que celui de l'algorithme de la fonction *ddl.m*). La commande est

```
cd C :/EF/exemples/Assemblage_elementaire, ...
    exemple1_matrice_elem_prepa_T ;
```

Les résultats sont ceux de l'exemple précédent.

Exemple 2 : Cet exemple traite, sur le modèle du fichier précédent *exemple1_matrice_elem_prepa_T*, le cas du maillage du carré par deux triangles de la figure 4.3-10 et des deux types d'espace, P_1^c et P_2^c , de la figure 4.3-11.

Le calcul effectué par *matrice_elem_prepa_T* dans ce fichier est aussi effectué dans le fichier *exemple2_matrice_elem_T* (cf. l'exemple 2 à la description de la fonction *matrice_elem_T*) pour la mise en œuvre de la fonction *matrice_elem_T*.

La commande est

```
cd C :/EF/exemples/Assemblage_elementaire, ...
    exemple2_matrice_elem_prepa_T ;
```

Algorithme :

EF/Assemblage_elementaire/matrice_elem_prepa_T.m

On remet ici la partie déclarative de la fonction :

```
function [Ass_Dom, SL_Dom, f_Dom]=matrice_elem_prepa_T(donnees,...
    i, T, Tc, X, Y, D, Ass_Dom, SL_Dom, f_Dom) ;
```

- Numéros des *ddl* du triangle numéro i :

```
E = D.E ;
```

```
for ic=1 :length(E)
```

```
    cpt{ic} = Tc{ic}(i, :) ;
```

```
end ;
```

- Coordonnées des sommets du triangle i :

```
XT = X(T(i,1 :3)) ;
```

```
YT = Y(T(i,1 :3)) ;
```

- Coefficients pour les calcul des fonctions de base et l'assemblage : (a1,a2), (b1,b2), (c1,c2) sont les coordonnées des 3 sommets du triangle, de le déterminant de la matrice Jacobienne (cf. la fonction *assemble_coeff_T*), Ac, Bc les coordonnées cartésiennes des points de la formule de quadrature, *coeff* les coefficients de la formule de quadrature et L1, L2, L3 les coordonnées barycentriques des points de la formule de quadrature :

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_prepa_T

```
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
    assemble_coeff_T(XT, YT, D) ;
```

- Calcul de la variable $\text{Ass\_Dom}\{i, j, k\}$ relative au triangle i , l'espace d'EF sur K de numéro $E(j)$ et la dérivation d'ordre $k - 1 = D.\text{der}(k)$:

```
for j=1 :length(E)%Boucle sur les types d'espaces utilises
    I{j} = pre_assemble_T([], E(j), [], a1, b1, c1, a2, b2, c2, ...
        de, Ac, Bc, coeff, 1, D, L1, L2, L3) ;
    for k=1 :length(D.der)%Les trois derivations 0,1,2
        %Calcul de Ass_Dom{i,j,k} avec pre_assemble_T
        Ass_Dom{i,j,k} = pre_assemble_T([], E(j), D.der(k), ...
            a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, 3, D, ...
                L1, L2, L3) ;
    end ;
end ;
```

- Calcul de SL_Dom , matrice annexe utilisée par exemple pour des calculs de norme. Le terme $\text{SL\_Dom}\{j, k, d1, d2\}$ calculé ci-dessous est la matrice élémentaire associée à la forme bilinéaire définie sur $F_k \times F_j$ par

$$\int_K u_{,d1} v_{,d2}.$$

Rappelons la relation (3.6) du paragraphe 3.1.2.1 dans le cas de la forme bilinéaire sur $E_2 \times E_1$ (en adaptant la dénomination des indices au cas ci-dessous) :

$$\int_K \omega u_{,d1} \tau_{,d2} \text{ donne } A_{jk} = \int_K \omega \psi_{k,d1} \phi_{j,d2} = (C^3 * \Omega^3 * \tau_{k2}) * u_{k1}, \quad (3.6)$$

qui correspond au calcul de $\text{SL\_Dom}\{1, 2, d1, d2\}$ sur $F_2 \times F_1$. En adaptant cette formule, on obtient, avec ici une valeur du poids $\omega = 1$, que la contribution du triangle i au terme $\text{SL\_Dom}\{j, k, d1, d2\}(\text{cpt}\{j\}, \text{cpt}\{k\})$ pour la forme bilinéaire sur $F_k \times F_j$ est alors donnée par la valeur (la ligne ci-dessous ne fait partie de l'algorithme) :

$$(\text{I}\{j\} * \text{Ass\_Dom}\{i, j, d2\}') * \text{Ass\_Dom}\{i, k, d1\}.$$

Pour traiter tous les cas d'espaces et de dérivation la boucle est alors la suivante :

```
for d1=1 :length(D.der) ;%Boucle sur les 3 types de derivations
    for d2=1 :length(D.der) ;%Boucle sur les 3 types de derivations
        for j=1 :1 :length(D.E)%Boucle sur les types d'espaces
            p1 = I{j}.*Ass_Dom{i,j,d2}' ;
```


4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_T

```

        for k=1 :length(D.E)%boucle sur les types d'espaces
            SL_Dom{j,k,d1,d2}( cpt{j},cpt{k} ) = ...
                SL_Dom{j,k,d1,d2}( cpt{j},cpt{k} )...
                    + p1*Ass_Dom{i,k,d1} ;
        end ;
    end ;
end ;

```

• Contribution du triangle i au calcul de la variable f_Dom , «graphe» des termes *a priori* non nuls des matrices d'assemblage :

```

for j=1 :1 :length(D.E)%Boucle sur les types d'espaces utilises
    for k=1 :length(D.E)%Boucle sur les types d'espaces utilises
        f_Dom{j,k}( cpt{j},cpt{k} ) = 1 ;
    end ;
end ;

```

Fonction :

```
function [Ass_Fr, SL_Fr, f_Fr] = matrice_elem_prepa_A(donnees, ...
    i, T, Tc, Pf, X, Y, D, Ass_Fr, SL_Fr, f_Fr) ;
```

Calcul de valeurs de fonctions de base, de matrices d'assemblage type norme d'énergie et graphe des matrices d'assemblage. La syntaxe est

```
[Ass_Fr, SL_Fr, f_Fr] = matrice_elem_prepa_A(donnees, ...
    I, T, Tc, Pf, X, Y, D, Ass_Fr, SL_Fr, f_Fr) ;
```

pour effectuer des calculs analogues à `matrice_elem_prepa_T`, mais relativement aux arêtes sur la frontière et à l'intégration sur ces arêtes. Par rapport à `matrice_elem_prepa_T`, cette fonction rajoute en entrée la variable `Pf` qui décrit les arêtes de la frontière dans une numérotation locale, sur la frontière, des arêtes. La variable i indique alors le numéro d'arête sur lequel on effectue les calculs. Les variables qui changent par rapport à `matrice_elem_prepa_T` sont les suivantes :

Pf : DOUBLE ARRAY. Matrice de taille (Nombre d'arêtes sur la frontière) $\times$ 4. Sur la frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur la frontière. Soit une arête de numéro j dans sa numérotation sur le morceau la frontière, `Pf(j,1)` est le numéro de triangle auquel appartient cette arête, `Pf(j,2)` son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3), `Pf(j,3)` son numéro global d'arête dans la numérotation globale des arêtes et `Pf(j,4)` le numéro de morceau de frontière auquel appartient cette arête. `Pfc` vaut [] si `Front` = []. Les variables `Pf` et `Pfc` contiennent la même information si l'on sait que `Pf(:,1:3) = [Pfc{1}; ...; Pfc{end}]`. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Ass_Fr : CELL ARRAY. Tableau de cellules $N \times N_E \times 3$ où N est le nombre de d'arêtes sur la frontière et $N_E = \text{length}(\text{donnees.D.E})$. Sur la frontière, les arêtes reçoivent une numérotation locale, l'arête numéro i est décrite par `Pf(i,:)`. L'ordre choisi est celui des arêtes de `Front{1}`, `Front{2}` etc. (cf. aussi les variables `Pf` et `Pfc`). Le terme `Ass_Fr{i,j,k}` est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K de numéro `Pf(i,1)` qui porte l'arête de numéro i dans la numérotation locale des arêtes sur la frontière, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par `donnees.D.E(j)`. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro `donnees.intA` utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace d'EF sur le triangle, l'ordre des fonctions de base étant celui donné au Paragraphe 2.3.3. Cette variable représente le champ `Ass_Fr` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

En entrée de cette fonction on peut poser `Ass_Fr` = [].

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_A

SL_Fr : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$.
 Analogue de **SL_Dom** relativement à la forme bilinéaire

$$\int_{\Gamma} v_{,k-1} w_{,m-1},$$

où Γ est la frontière de Ω . Cette variable représente le champ **SL_Fr** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

F_Fr : CELL ARRAY. Analogue de **f_Dom**, la mesure est ici celle des arêtes frontières.

Exemple 1 : Cet exemple est donné par la commande

```
cd C :/EF/exemples/Assemblage_elementaire, ...
                                exemple1_matrice_elem_prepa_A ;
```

équivalent pour les arêtes des Exemples 1 et 1 *bis* vus à la description de la fonction **matrice_elem_prepa_T**. Dans cet exemple le maillage est donc constitué du triangle K de sommets $(0,0)$, $(1,0)$, $(0,1)$ et $\{\Phi_i\}_{1 \leq i \leq 3}$ désigne la base de $E_1 = P_1^c$ et $\{\Psi_i\}_{1 \leq i \leq 6}$ celle de $E_2 = P_2^c$. Notons $m \in [\mathbb{R}^2]^{15}$ le vecteur dont les composantes sont, dans l'ordre choisi dans le code, les coordonnées des points de la formule de quadrature sur les arêtes du triangle. La numérotation de ces points suit le même ordre que celui représenté à la Figure 4.3 – 9 ci-dessus, dans le cad d'un triangle de sommets $(0,0)$, $(1/2,0)$, $(0,1)$ (*cf.* description de la fonction **assemble_coeff_A**). Par commodité nous redonnons ci-dessous cette figure.

Points de la formule de quadrature dans leur ordre de numérotation

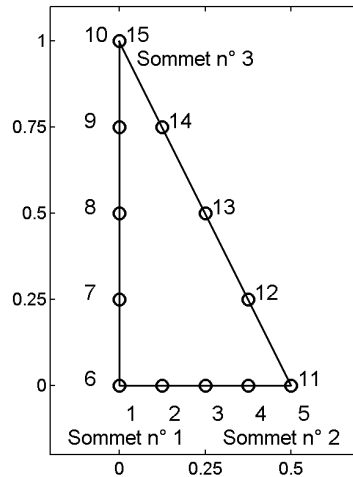


Figure 4.3-9 (figure donnée à la description de la fonction **assemble_coeff_A**).

Avec ce choix de T_c (*cf.* Paragraphe 2.3), les *ddl* de E_1 sont successivement les 3 sommets du triangle K , les *ddl* de E_2 sont successivement les 3 sommets du triangles, puis les milieux des sommets 1 et 2, des sommets 1 et 3, et des sommets 2 et 3 du triangle K .

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_prepa_A

Voici ce qu'on trouve pour les exemples suivants de fonctions de base et de dérivations :

$$\begin{aligned}\Phi_1(m) &= \text{Ass\_Fr}\{1, 1, 1\}(:, 1) = (1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 0, 0, 0, 0, 0)' , \\ \Phi_2(m) &= \text{Ass\_Fr}\{1, 1, 1\}(:, 2) = (0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, 0, 0, 0, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0)' , \\ \Phi_3(m) &= \text{Ass\_Fr}\{1, 1, 1\}(:, 3) = (0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1)' .\end{aligned}$$

Ensuite, voici un exemple de calcul des matrices associées aux formes bilinéaires sur $E_1 \times E_1$ et qui sont stockées dans $\text{SL\_Fr}\{1, 1, :, :\}$ (à titre de vérification, on pourra refaire les calculs comme indiqué dans le premier exemple de la description de la fonction `matrice_elem_A`, dans le cas de deux triangles) :

$$\int_{\delta K} \Phi_j \Phi_i = \text{SL\_Fr}\{1, 1, 1, 1\}(\mathbf{i}, \mathbf{j}) = \begin{pmatrix} 2/3 & 1/6 & 1/6 \\ 1/6 & 1/3 + \sqrt{2}/3 & \sqrt{2}/6 \\ 1/6 & \sqrt{2}/6 & 1/3 + \sqrt{2}/3 \end{pmatrix}.$$

On vérifie que la somme des termes de la matrice est égale au périmètre du triangle K *i.e.* $2 + \sqrt{2}$. Cela est du que fait que cette somme est égale à

$$\sum_{1 \leq i, j \leq 3} \int_{\delta K} \Phi_j \Phi_i = \int_{\delta K} (\Phi_1 + \Phi_2 + \Phi_3)^2 = \int_{\delta K} 1^2 = 2 + \sqrt{2}.$$

Termes faisant intervenir des dérivations :

$$\begin{aligned}\int_{\delta K} \Phi_{j,x} \Phi_{i,y} &= \text{SL\_Fr}\{1, 1, 1, 2\}(\mathbf{i}, \mathbf{j}) \\ &= \begin{pmatrix} -1 & -1/2 - \sqrt{2}/2 & -1/2 - \sqrt{2}/2 \\ 1 & 1/2 + \sqrt{2}/2 & 1/2 + \sqrt{2}/2 \\ 0 & 0 & 0 \end{pmatrix} = \text{SL\_Fr}\{1, 1, 2, 1\}(\mathbf{j}, \mathbf{i}).\end{aligned}$$

On vérifie que la somme des termes de la matrice est nulle. Cela est du au fait que cette somme est égale à

$$\sum_{1 \leq i, j \leq 3} \int_{\delta K} \Phi_{j,x} \Phi_{i,y} = \int_{\delta K} (\Phi_1 + \Phi_2 + \Phi_3)_{,x} (\Phi_1 + \Phi_2 + \Phi_3)_{,y} = \int_{\delta K} 1_{,x} 1_{,y} = 0.$$

Ensuite, exemple de calcul des matrices associées aux formes bilinéaire sur $E_2 \times E_1$ stockées dans $\text{SL\_Fr}\{1, 2, :, :\}$:

$$\begin{aligned}\int_{\delta K} \Psi_j \Phi_i &= \text{SL\_Fr}\{1, 2, 1, 1\}(\mathbf{i}, \mathbf{j}) \\ &= \begin{pmatrix} 1/3 & 0 & 0 & 1/3 & 1/3 & 0 \\ 0 & 1/6 + \sqrt{2}/6 & 0 & 1/3 & 0 & \sqrt{2}/3 \\ 0 & 0 & 1/6 + \sqrt{2}/6 & 0 & 1/3 & \sqrt{2}/3 \end{pmatrix} \\ &= \text{SL\_Fr}\{2, 1, 1, 1\}(\mathbf{j}, \mathbf{i}) \text{ (par symétrie de la forme bilinéaire).}\end{aligned}$$

4.3. DESCRIPTION DES F^{NS} F^N Assemblage_elementaire/matrice_elem_prepa_A

On vérifie aussi comme pour le cas du P_1 que la somme des termes de cette matrice est égale au périmètre du triangle K :

$$\sum_{1 \leq j \leq 6} \left(\sum_{1 \leq i \leq 3} \int_{\delta K} \Psi_j \Phi_i \right) = \int_{\delta K} (\Psi_1 + \Psi_2 + \Psi_3 + \Psi_4 + \Psi_5 + \Psi_6)(\Phi_1 + \Phi_2 + \Phi_3) = \int_{\delta K} 1^2 = 2 + \sqrt{2}.$$

Exemple 2 : Cet exemple est l'équivalent pour les arêtes frontières de l'exemple `exemple2_matrice_elem_prepa_T` de la fonction précédente avec le maillage du carré par deux triangles de la figure 4.3-10 et des deux types d'espace, P_1^c et P_2^c , de la figure 4.3-11.

La commande est

```
cd C :/EF/exemples/Assemblage_elementaire, ...
    exemple2_matrice_elem_prepa_A ;
```

Algorithme.

L'algorithme suit le même principe pour les boucles que celui utilisé dans la fonction `matrice_elem_A`, le calcul des variables de sortie se fait suivant le principe de calcul de la fonction `matrice_elem_prepa_T` adapté au cas des arêtes frontières du maillage (voir la description de l'algorithme de ces deux fonctions). Pour plus de détails on pourra se référer au *script* de cette fonction.

Fonction :

```
function [Ass_Ad, SL_Ad, f_Ad]=matrice_elem_prepa_Ad(donnees, ...
```

```
    i, T, Tc, Pa, X, Y, D, Ass_Ad, SL_Ad, f_Ad) ;
```

Calcul de valeurs de fonctions de base, de matrices d'assemblage type norme d'énergie et graphe des matrices d'assemblage. La syntaxe est

```
[Ass_Ad, SL_Ad, f_Ad]=matrice_elem_prepa_Ad(donnees, ...
```

```
    i, T, Tc, Pa, X, Y, D, Ass_Ad, SL_Ad, f_Ad) ;
```

pour effectuer des calculs analogues à `matrice_elem_prepa_T`, mais relativement aux arêtes et à l'intégration sur ces arêtes. Par rapport à la fonction `matrice_elem_prepa_T`, cette fonction rajoute en entrée la variable `Pa` qui décrit les arêtes du maillage dans la numérotation globale des arêtes du maillage. La variable `i` indique alors le numéro d'arête sur lequel on effectue les calculs. Les variables qui changent par rapport à `matrice_elem_prepa_T` sont les suivantes :

Pa : DOUBLE ARRAY. Matrice de taille (Nombre d'arête du maillage)×5. Description des arêtes du maillage tel que `Pa(j, 1)` est un numéro de triangle auquel appartient l'arête `j` et `Pa(j, 2)` son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour `Pa(j, 3 : 4)` si l'arête appartient à un deuxième triangle sinon `Pa(j, 3)` et `Pa(j, 4)` sont nuls (arête sur la frontière). Enfin `Pa(j, 5)` est le numéro du morceau de frontière si l'arête `j` est sur la frontière, nul sinon ou si on a posé `Front = []`. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Ass_Ad : CELL ARRAY. Tableau de cellules $N \times N_E \times 3 \times 2$ où N est le nombre d'arêtes et $N_E = \text{length}(\text{donnees.D.E})$. Le terme `Ass_Ad{i, j, k, m}` est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K_m de numéro `Pa(i, 1)` pour $m = 1$ et `Pa(i, 3)` pour $m = 2$ pour l'arête de numéro `i` dans la numérotation globale des arêtes, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par `donnees.D.E(j)`. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Pour $m = 1$, on obtient le résultat concernant la restriction des fonctions de base sur le premier triangle auquel appartient l'arête `i` et de numéro donné par `Pa(i, 1)`, pour $m = 2$, le résultat concerne l'autre éventuel triangle de numéro `Pa(i, 3)`, s'il n'y a pas de second triangle on pose `Ass_Dom{i, j, k, m} = []`. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro `donnees.intA` utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ `Ass_Ad` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

En entrée de cette fonction on peut poser `Ass_Ad = []`.

SL_Ad : CELL ARRAY. Tableau $N_E \times N_E \times 3 \times 3$, $N_E = \text{length}(\text{donnees.D.E})$. Analogue de `SL_Dom` relativement à la forme bilinéaire

$$\int_{\Gamma} v_{,k-1} w_{,m-1} + \frac{1}{2} \int_{\Sigma_1} v_{,k-1} w_{,m-1} + \frac{1}{2} \int_{\Sigma_2} v_{,k-1} w_{,m-1},$$

4.3. DESCRIPTION DES $F^{NS} F^N$ `Assemblage_elementaire/matrice_elem_prepa_Ad`

avec Σ_p , $p = 1$ ou 2 réunion des arêtes intérieures, $p = 1$ signifiant que, sur une arête i donnée, on considère la restriction de $v_{k-1}w_{m-1}$ au triangle de numéro $\text{Pa}(i, 1)$ du Tableau 2.2.3-1b et pour $p = 2$ au triangle de numéro $\text{Pa}(i, 3)$. Cette variable représente le champ `SL_Ad` du Tableau 2.2.3-1a, Paragraphe 2.2.3.

`F_Ad` : CELL ARRAY. Analogue de `f_Dom`, la mesure est ici celle des arêtes du domaine.

Exemple 1 : Cet exemple est l'équivalent pour les arêtes de l'exemple `exemple1_matrice_elem_prepa_A`, avec la commande

`exemple1_matrice_elem_prepa_Ad`

Ici les arêtes du maillage sont aussi celles de la frontière, on remarque cependant que l'ordre des arêtes frontière donné par `Pf` diffère de celui des arêtes globales donné par `Pa`. Cependant, les résultats donnés par `matrice_elem_prepa_Ad` sont identiques à ceux de `matrice_elem_prepa_A`, car il n'y donc pas d'arêtes intérieures et car l'ordre des arêtes n'intervient pas dans les résultats.

Exemple 2 : Cet exemple est l'équivalent pour les arêtes de l'exemple `exemple2_matrice_elem_prepa_A` de la fonction précédente avec le maillage du carré par deux triangles de la figure 4.3-10 que nous remettons ici par commodité comme les figures qui suivent :

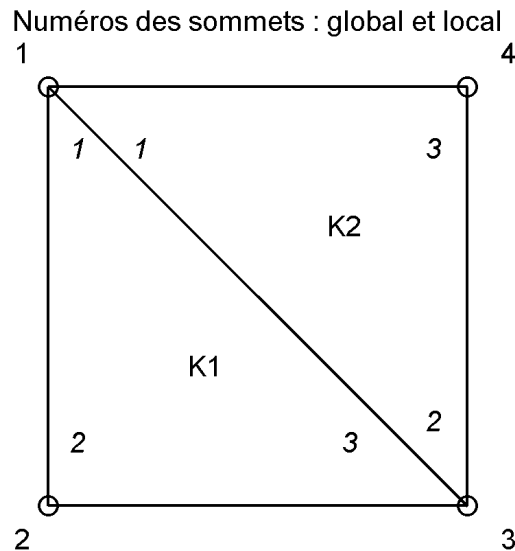


Figure 4.3-10. Numérotation des sommets du maillages (chiffres de 1 à 4 en caractères droits) et numérotation locale des triangles (chiffres en italique) (figure données à la description de la fonction `matrice_elem_T`).

Ce maillage est munis des deux types d'espace, P_1^c et P_2^c , dont les *ddl* sont donnés

4. DESCRIPTION DU CODE `FN Assemblage_elementaire/matrice_elem_prepa_Ad`
à la figure 4.3-11 :

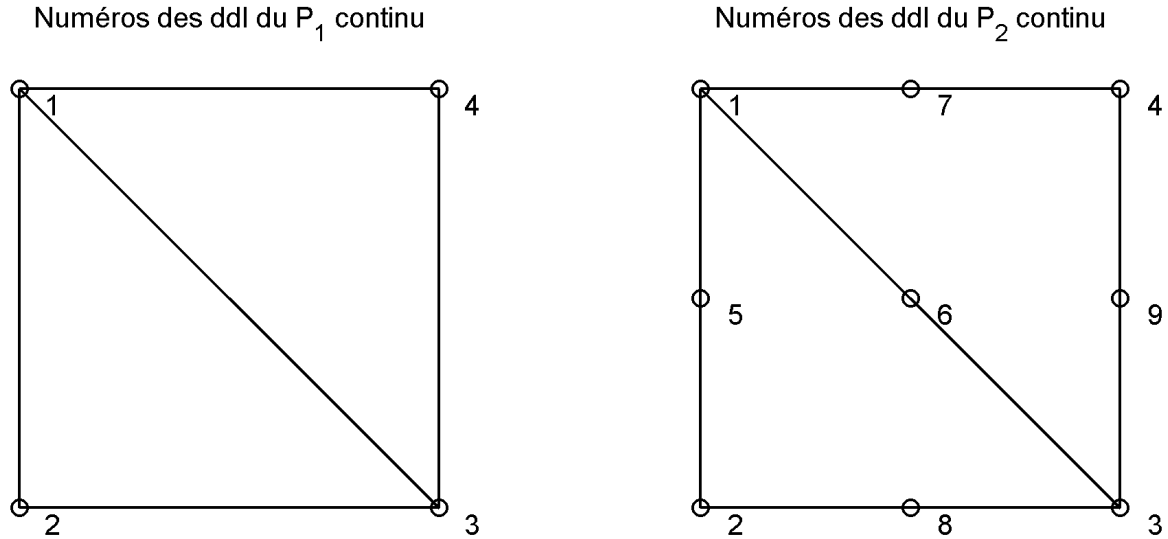


Figure 4.3-11. Degrés de liberté du P_1 continu et du P_2 continu (figures données à la description de la fonction `matrice_elem.T`).

Ensuite, la figure 4.3-13 donne la numérotation globale des arêtes :

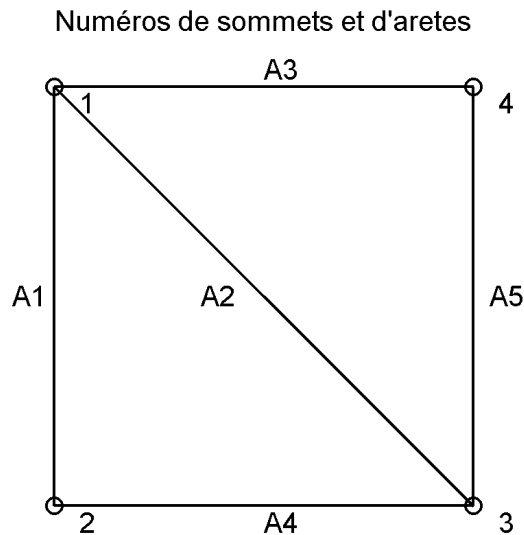


Figure 4.3-13. Numérotation globale des arêtes (figure donnée à la description de la fonction `matrice_elem.Ad`).

La numérotation des points de la formule de quadrature en fonction des numéros

4.3. DESCRIPTION DES $F^{NS} F^N$ Assemblage_elementaire/matrice_elem_prepa_Ad

de sommets locaux du triangle est rappelée ci-dessous :

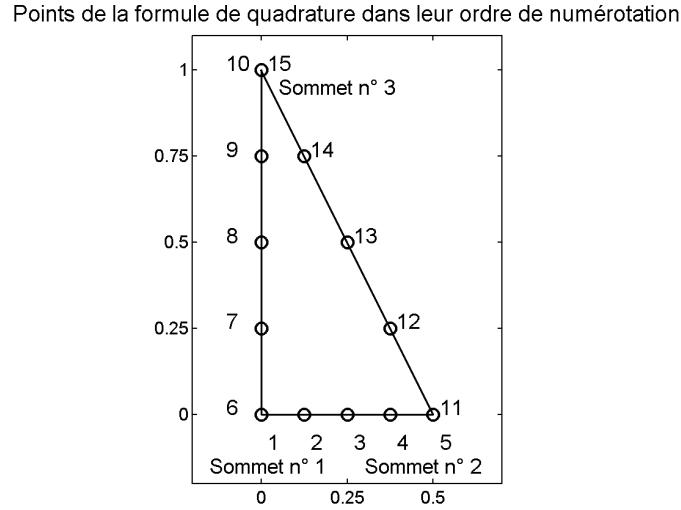


Figure 4.3-9 (figure donnée à la description de la fonction `assemble_coef_A`).

Ainsi, sur le triangle K_1 de sommets respectifs, locaux et globaux, 1, 2, 3, la numérotation des points de la formule de quadrature se fait du sommet global 1 vers le sommet global 2, puis du sommet global 2 vers le sommet global 3 et enfin du sommet global 3 vers le sommet global 3.

Ensuite, sur le triangle K_2 de sommets respectifs, locaux 1, 2, 3 et globaux 1, 3, 4, la numérotation des points de la formule de quadrature se fait du sommet global 1 vers le sommet global 3, puis du sommet global 1 vers le sommet global 4, et enfin du sommet global 3 vers le sommet global 4.

La commande est

```
cd C :/EF/exemples/Assemblage_elementaire, ...
    exemple2_matrice_elem_prepa_Ad ;
```

L'arête intérieure est donc l'arête numéro 2. Ainsi les matrices $\text{Ass\_Ad}\{2, 1, 1, 1\}$ et $\text{Ass\_Ad}\{2, 1, 1, 2\}$ donnent, pour le P_1 continu, les valeurs des fonctions de base locales $(\phi_i^2)_{1 \leq i \leq 3}$ sur le triangle numéro $\text{Pa}(2, 1) = 2$ i.e. K_2 et $(\phi_i^1)_{1 \leq i \leq 3}$ sur le triangle numéro $\text{Pa}(2, 3) = 1$, i.e. K_1 , des formules de quadrature sur les arêtes de ces triangles.

Pour $\text{Ass\_Ad}\{2, 1, 1, 1\}$ (triangle K_2) on a :

$$\begin{aligned} \phi_1^2(m) &= \text{Ass\_Ad}\{2, 1, 1, 1\}(:, 1) = (1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 0, 0, 0, 0, 0)' , \\ \phi_2^2(m) &= \text{Ass\_Ad}\{2, 1, 1, 1\}(:, 2) = (0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, 0, 0, 0, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0)' , \\ \phi_3^2(m) &= \text{Ass\_Ad}\{2, 1, 1, 1\}(:, 3) = (0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1)' , \\ \phi_{1,x}^2(m) &= \text{Ass\_Ad}\{2, 1, 2, 1\}(:, 1) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{2,x}^2(m) &= \text{Ass\_Ad}\{2, 1, 2, 1\}(:, 2) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' , \end{aligned}$$

4. DESCRIPTION DU CODE F^N `Assemblage_elementaire/matrice_elem_prepa_Ad`

$$\begin{aligned}\phi_{3,x}^2(m) &= \text{Ass\_Ad}\{2, 1, 2, 1\}(:, 3) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{1,y}^2(m) &= \text{Ass\_Ad}\{2, 1, 3, 1\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' , \\ \phi_{2,y}^2(m) &= \text{Ass\_Ad}\{2, 1, 3, 1\}(:, 2) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{3,y}^2(m) &= \text{Ass\_Ad}\{2, 1, 3, 1\}(:, 3) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' \end{aligned}$$

et pour $\text{Ass\_Ad}\{2, 1, 1, 2\}$ (triangle K_1) on a :

$$\begin{aligned}\phi_1^1(m) &= \text{Ass\_Ad}\{2, 1, 1, 2\}(:, 1) = (1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0, 0, 0, 0, 0)' , \\ \phi_2^1(m) &= \text{Ass\_Ad}\{2, 1, 1, 2\}(:, 2) = (0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, 0, 0, 0, 0, 1, \frac{3}{4}, \frac{1}{2}, \frac{1}{4}, 0)' , \\ \phi_3^1(m) &= \text{Ass\_Ad}\{2, 1, 1, 2\}(:, 3) = (0, 0, 0, 0, 0, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1)' , \\ \phi_{1,x}^1(m) &= \text{Ass\_Ad}\{2, 1, 2, 2\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' , \\ \phi_{2,x}^1(m) &= \text{Ass\_Ad}\{2, 1, 2, 2\}(:, 2) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{3,x}^1(m) &= \text{Ass\_Ad}\{2, 1, 2, 2\}(:, 3) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{1,y}^1(m) &= \text{Ass\_Ad}\{2, 1, 3, 2\}(:, 1) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{2,y}^1(m) &= \text{Ass\_Ad}\{2, 1, 3, 2\}(:, 2) = -(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{3,y}^1(m) &= \text{Ass\_Ad}\{2, 1, 3, 2\}(:, 3) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' . \end{aligned}$$

On remarque que, par construction $\text{Ass\_Ad}\{2, 1, 1, 1\}$ et $\text{Ass\_Ad}\{2, 1, 1, 2\}$ sont identiques.

Si on regarde l'arête numéro 1 située sur la frontière, on a $\text{Pa}(1, 1) = 1$ et $\text{Pa}(1, 3) = 0$, ainsi $\text{Ass\_Ad}\{1, 1, 1, 1\}$ représente les valeurs des fonctions de base aux points de la formule de quadrature du triangle 1 et prend donc la même valeur que $\text{Ass\_Ad}\{2, 1, 1, 2\}$ détaillée ci-dessus. Ensuite comme cette arête n'a pas de second triangle d'appartenance on a $\text{Ass\_Ad}\{1, 1, 1, 2\} = []$.

De même, pour le P_2 continu les valeurs de Ass_Ad relatives à l'arête numéro 2 sont donnée par $\text{Ass\_Ad}\{2, 2, 1, 1\}$ pour le triangle K_2 et par $\text{Ass\_Ad}\{2, 2, 1, 2\}$ pour le triangle K_1 . C'est le second indice de Ass_Ad , de valeur 2 ici, qui indique qu'il s'agit du second des types d'espaces d'EF utilisés.

Algorithme.

L'algorithme suit le même principe pour les boucles que celui utilisé dans la fonction `matrice_elem_Ad`, le calcul des variables de sortie se fait suivant le principe de calcul de la fonction `matrice_elem_prepa.T` adapté au cas des arêtes du maillage (voir la description de l'algorithme de ces deux fonctions). Pour plus de détails on pourra se référer au *script* de cette fonction.

Fonction :

```
function [Sc_Ad] = matrice_elem_Ad_int_LR(donnees, i, T, Tc, ...
```

```
    Pa, X, Y, Uc, D, Mat_syst_Ad_int_LR, It, Ass_Ad, Sc_Ad) ;
```

Calcul de matrices élémentaires pour l'assemblage de la méthode de Lesaint-Raviart. C'est la partie en a_{LR} de la forme bilinéaire $\mathcal{A}$ dans (1.5).

Cette fonction permet l'assemblage sur les arêtes intérieures du domaine du membre du terme de saut provenant de la méthode de Lesaint-Raviart pour les EF discontinus. Elle prend aussi en compte les EF continus du code avec un saut nul. L'assemblage n'est réalisé qu'à l'ordre 0 en dérivation. Il est à noter que cette fonction ne prend en compte que les contributions des arêtes intérieures.

Elle effectue le calcul d'une matrice élémentaire associée à la méthode de Lesaint-Raviart du système d'EDP et ajoute la contribution de cette matrice à la matrice globale de type `CELL` du système qui contient les contributions aux arêtes du domaine. Le calcul est effectué sur les arêtes frontières du maillage du domaine.

Le type de dérivation pris en compte est définie par la forme bilinéaire suivante définie au Paragraphe 1.2 relation (1.17) par :

$$\int_{\Sigma} a_{LR}(x, X^0, X, Y) = \sum_{1 \leq j \leq n_{LR}} \sum_{i \in M_j} \int_{\Sigma} (X_i^{+, u_j}(x) - X_i^{-, u_j}(x)) Y_i^{+, u_j}(x) |u_j(x, X^0(x)) \cdot n(x)| \quad (1.17)$$

avec $n(x)$ est une des deux normales unitaires à Σ au point $x \in \Sigma$, les termes $u_j(\cdot, X^0)$, $1 \leq j \leq n_{LR}$ sont une famille de fonction de $\bar{\Omega}$ à valeur dans $\mathbb{R}^2$, elles indiquent chacune une direction de dérivation, et enfin M_j un ensemble d'entiers compris entre 1 et n qui indique que l'on dérive les inconnues $i \in M_j$ suivant u_j . Pour une fonction f définie sur Ω on définit $f^{\pm, u}$ par :

$$f^{\pm, u}(x) = \lim_{\epsilon \rightarrow 0^{\pm}} f(x + \epsilon u(x)). \quad (1.18)$$

On remarque que la dérivation n'intervient que sur l'inconnue numéro i de la ligne i , comme par exemple dans le traitement du transport vu au Paragraphe 2.7, avec l'utilisation, aux sous-Paragraphe 2.7.1 et 2.7.1.3 de la méthode de Lesaint-Raviart pour ce type de problème ou aussi dans l'étude des écoulements de fluides viscoélastiques vus au Paragraphe 2.11.8. Pour la prise en compte d'autres inconnues, si le problème s'y prête, on pourra soit modifier l'ordre des équations, soit calculer à part les matrice d'assemblage supplémentaires avec la fonction `matrice_elem_Ad_int_LR`. Sinon, on pourra construire une nouvelle fonction en adaptant la fonction `matrice_elem_Ad_int_LR` au problème.

L'appel

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

```
[Sc_Ad] = matrice_elem_Ad_int_LR(donnees, i, T, Tc, ...
```

```
Pa, X, Y, Uc, D, Mat_syst_Ad_int_LR, It, Ass_Ad, Sc_Ad) ;
```

permet de calculer la matrice élémentaire, par blocs, associée à l'arête i situé sur les triangle $\text{Pa}(i, 1)$, $\text{Pa}(i, 3)$, lorsqu'il s'agit d'une arête intérieure, et l'ajoute à la matrice globale du système relative à l'EDP et aux arêtes, désigné ci-dessus par **Sc_Ad**, matrice de type **CELL**.

La variable **donnees** est la variable de type **struct** qui représente les données du problème. La variable **T** est le tableau qui décrit le maillage, **Tc** représente les *ddl* des différents types d'espaces, **Pa** est un tableau décrivant les arêtes, **X** et **Y** les coordonnées en x et en y des sommets du maillage, **Uc** est un tableau de cellules représentant un élément de l'espace EF produit E comme par exemple un itéré de la solution dans un point fixe. Le terme **Mat_syst_Ad_int_LR** est un **handle** (ou un nom de fonction) décrivant le champ de vecteur suivant lequel on dérive sur les arêtes frontières. La variable **Ass_Ad** contient les valeurs des fonctions de base ainsi que de leurs dérivées au point de la formule de quadrature sur l'arête courante. En général cette variable est calculée par **EF_prepa** avec la fonction **matrice_elem_prep_Ad**, ou directement comme dans le deuxième des deux exemples qui suivent. La variable **Sc_Ad** est un tableau de type **CELL** qui représente la matrice globale d'assemblage. **It** est le vecteur d'entiers qui indiquent sur quelles composantes on effectue la dérivation.

Par rapport à la formule (1.17) remise ci-dessus, c'est la partie

$$\sum_{i \in M_j} \int_{\Sigma} (X_i^{+, u_j}(x) - X_i^{-, u_j}(x)) Y_i^{+, u_j}(x) |u_j(x, X^0(x)) \cdot n(x)|$$

qui est assemblée, **It** désignant l'ensemble M_j . Dans (1.17) la sommation en j est effectuée dans la fonction d'assemblage global **EF_assemblage**, avec la variable de boucle **iD** = j . La boucle est réalisée sur les lignes de la variable **D_LR** (cf. Tableau 2.2.1-1d, Paragraphe 2.2.1) qui est la variable de type **CELL** telle que **D_LR{iD, 1}** et **D_LR{iD, 2}** donnent successivement les valeurs de **Mat_syst_Ad_int_LR** et de **It** qui définissent le champ u_j et l'ensemble M_j ci-dessus.

Variables d'entrée :

Les variables sont identiques à celles de la fonction **matrice_elem_T** excepté les variables décrites ci-dessous **Mat_syst_Ad_int_LR**, **Ass_Ad**, **Sc_Ad** à la place de **Mat_syst_Dom**, **Ass_Dom**, **Sc_Dom** dans **matrice_elem_T**, la variable **Pa** qui décrit les arêtes et enfin la variable **It** qui indique les composantes sur lesquelles on dérive. Il est à noter que les calculs de matrices élémentaires sont effectués pour toutes les composantes de l'espace EF produit E . Les contributions de ces matrices ajoutées à la matrice globale sont celles indiquées par la variable **It**. Précisons ces variables.

Pa : **DOUBLE ARRAY**. Matrice de taille (Nombre d'arête du maillage) $\times$ 5. Description des arêtes du maillage tel que **Pa(j, 1)** est un numéro de triangle auquel appartient

4.3. DESCRIPTION DES F^{NSFN} `Assemblage_elementaire/matrice_elem_Ad_int_LR`

l'arête j et $\text{Pa}(j, 2)$ son numéro local d'arête sur ce triangle (1, 2 ou 3). *Idem* pour $\text{Pa}(j, 3 : 4)$ si l'arête appartient à un deuxième triangle sinon $\text{Pa}(j, 3)$ et $\text{Pa}(j, 4)$ sont nuls (arête sur la frontière). Enfin $\text{Pa}(j, 5)$ est le numéro du morceau de frontière si l'arête j est sur la frontière, nul sinon ou si on a posé `Front = []`. Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Ass_Ad : CELL ARRAY. Tableau de cellules $N \times N_E \times 3 \times 2$ où N est le nombre d'arêtes et $N_E = \text{length}(\text{donnees.D.E})$. Le terme $\text{Ass\_Ad}\{i, j, k, m\}$ est le tableau des valeurs, aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle K_m de numéro $\text{Pa}(i, 1)$ pour $m = 1$ et $\text{Pa}(i, 3)$ pour $m = 2$ pour l'arête de numéro i dans la numérotation globale des arêtes, de la dérivée d'ordre $k - 1$ des fonctions de base de l'espace d'EF E_K de type $\mathcal{P}_{p,q}^d(K)$ ou $\mathcal{P}_{p,q}^c(K)$ du Paragraphe 2.3.3, de numéro donné par $\text{donnees.D.E}(j)$. Ces fonctions de base sont aussi celles de l'espace vectoriel $\mathcal{P}_{p,q}(K)$, dans l'ordre donné au Paragraphe 2.3.3. Pour $m = 1$, on obtient le résultat concernant la restriction des fonctions de base sur le premier triangle auquel appartient l'arête i et de numéro donné par $\text{Pa}(i, 1)$, pour $m = 2$, le résultat concerne l'autre éventuel triangle de numéro $\text{Pa}(i, 3)$, s'il n'y a pas de second triangle on pose $\text{Ass\_Dom}\{i, j, k, m\} = []$. Le nombre de lignes de ce tableau de valeurs est égal au nombre de points d'intégration de la formule de quadrature de numéro `donnees.intA` utilisée sur les arêtes et le nombre de colonnes à la dimension de l'espace vectoriel $\mathcal{P}_{p,q}(K)$. Cette variable représente le champ **Ass_Ad** du Tableau 2.2.3-1a, Paragraphe 2.2.3.

Sc_Ad : CELL ARRAY. Matrice de taille $n \times n$, n nombre de termes de l'espace EF produit E . Analogue de **Sc_Fr** pour l'assemblage dans la forme bilinéaire $\mathcal{A}$ de (1.5), Paragraphe 1.2, de la partie en $\bar{\Sigma}$ (réunion de toutes les arêtes du maillage) avec l'intégration du terme en $a_{\bar{\Sigma}}$. A ce terme s'ajoute la contribution du terme en a_{LR} de la méthode de Lesaint-Raviart. La forme bilinéaire $\bar{\Sigma}$ est définie par le champ `donnees.feq_Ad` et l'assemblage se fait par la fonction `matrice_elem_Ad`. Pour la méthode de Lesaint-Raviart, la forme bilinéaire est définie par le champ `donnees.D_LR` et l'assemblage se fait par la fonction `matrice_elem_Ad_int_LR`. La partie en $a_{\bar{\Sigma}}$ pourra être modifiée par l'utilisateur suivant le problème à traiter, ceci par le biais d'une fonction analogue à celle de `matrice_elem_Ad` à insérer dans une boucle du même type que celle effectuée sur les arêtes dans la fonction `EF_assemblage`. Pour la méthode de Lesaint-Raviart telle que programmée actuellement, la dérivation porte sur les inconnues principales et seules les cellules sur la diagonale de **Sc_Ad**, sont *a priori* non nuls.

Mat_syst_Ad_int_LR : FUNCTION_HANDLE ARRAY. Handle sur la fonction qui définit les équations du premier membre sur $\bar{\Sigma}$ et qui correspond au terme a_{LR} qui intervient dans la forme bilinéaire $\mathcal{A}$ de (1.5). La fonction définit alors le champ de vecteur de $\mathbb{R}^2$ suivant lequel on dérive. En pratique cette fonction est désignée par la variable `D_LR` décrite au Tableau 2.2.1-1d du Paragraphe 2.2.1. Les n_{LR} fonctions de (1.17) sont données successivement par `donnees.D_LR\{1, 1\}`, `donnees.D_LR\{2, 1\}`, $\dots$, `donnees.D_LR\{n_{LR}, 1\}`. La syntaxe de la fonction désignée par **Mat_syst_Ad_int_LR** est celle de la fonction donnée au Paragraphe 2.2.4 et au Tableau 2.2.4-1 de ce Paragraphe (*cf.* le cas de la méthode de Lesaint-Raviart).

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

La syntaxe de la fonction désignée par `Mat_syst_Ad_int_LR`, que nous notons ci-dessous `eq_LR`, est la suivante :

Syntaxe de la fonction appelée par `Mat_syst_Ad_int_LR`.

```
function [Mc,Nc] = eq_LR(donnees, numF, uc, a1, b1, c1, a2, b2, ...
                        c2, de, Ac, Bc, numK, numALoc, numA) ;
```

Les variables d'entrée sont identiques à celles de la fonction appelée par `matrice_elem_Ad`. Nous précisons les variables de sortie ci-dessous.

Variables de sortie :

`Mc, Nc` : CELL ARRAY. Vecteurs lignes de type double représentant les valeurs aux points de coordonnées données par `Ac` et `Bc` de chaque composante du champ suivant lequel on dérive.

Suite de la description des variables d'entrée de `matrice_elem_A_scm` :

`It` : DOUBLE. Indique les espaces sur lesquels on applique la méthode. En pratique la valeur de `It` s'obtient aussi à partir de `D_LR`. Les n_{LR} ensemble M_j de (1.17) sont donnés par les vecteurs `donnees.D_LR{1,2}`, `donnees.D_LR{2,2}`, ..., `donnees.D_LR{n_LR,2}`, associés chacun à leurs champs de vecteurs respectifs donnés par les `handles` de fonction `donnees.D_LR{1,1}`, `donnees.D_LR{2,1}`, ..., `donnees.D_LR{n_LR,1}`.

Variables de sortie :

`Sc_Ad` : CELL ARRAY. Matrice en entrée de la fonction augmentée de la contribution de a_{LR} .

Exemples.

Une utilisation de ce type de fonction est donnée au Paragraphe 2.7 qui traite d'un problème de transport, avec une présentation de la méthode de Lesaint-Raviart au Paragraphe 2.7.1 et une mise en œuvre au sous-Paragraphe 2.7.1.3 de la méthode, cette méthode fonction est aussi utilisée au Paragraphe 2.11.8 pour l'étude d'écoulements de type viscoélastiques.

Exemple 1 : Cet exemple est donné par la commande

```
cd C :/EF/exemples/Assemblage_elementaire, ...
```

```
[Ass_Ad, Sc_Ad, A] = exemple1_matrice_elem_Ad_int_LR
```

Dans cet exemple, on regarde le calcul de la matrice d'assemblage du terme

$$\int_{\Sigma} (f^+ - f^-) g^- |n.u| ds$$

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

sur le carré unité avec la structure de maillage décrite à la Figure 4.3-10, associé à l'espace EF produit $E = P_0^d \times P_1^d \times P_2^d$ et avec le champ suivant lequel on dérive donné par $u(x, y) = (1, 1)$. Nous remettons ci-dessous par commodité le maillage :

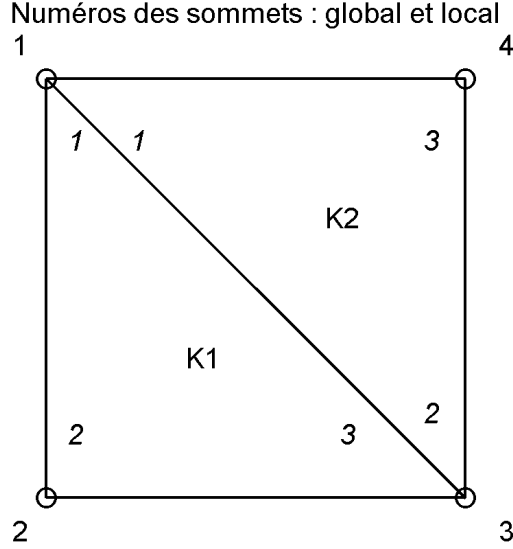


Figure 4.3-10. Numérotation des sommets du maillages (chiffres de 1 à 4 en caractères droits) et numérotation locale des triangles (chiffres en italique) (figure données à la description de la fonction `matrice_elem_T`).

On considère le cas d'une approximation P_0 -discontinue. Les *ddl* de $E = P_0^d$ sont représentés à la Figure 4.3-15 ci-dessous. Les fonctions de base de l'espace attachées à ces *ddl* sont notées ϕ_1 et ϕ_2 qui sont respectivement les fonctions indicatrices des triangles 1 et 2, le premier triangle étant celui de gauche (*cf.* Figure 4.3-10).

L'ensemble Σ des arêtes intérieures du maillage est donc le segment de longueur $\sqrt{2}$ commun aux deux triangles et sur ce segment on a $|n.u| = \sqrt{2}$, $n = (1/\sqrt{2}, 1/\sqrt{2})$ étant une des deux normales à ce segment.

On a alors pour la matrice d'assemblage

$$\begin{aligned} a_{11} &= \int_{\Sigma} (\phi_1^+ - \phi_1^-) \phi_1^+ |n.u| ds = \int_{\Sigma} (0 - 1) 0 |n.u| ds = 0, \\ a_{12} &= \int_{\Sigma} (\phi_2^+ - \phi_2^-) \phi_1^+ |n.u| ds = \int_{\Sigma} (1 - 0) 0 |n.u| ds = 0, \\ a_{21} &= \int_{\Sigma} (\phi_1^+ - \phi_1^-) \phi_2^+ |n.u| ds = \int_{\Sigma} (0 - 1) 1 |n.u| ds = -2, \\ a_{22} &= \int_{\Sigma} (\phi_2^+ - \phi_2^-) \phi_2^+ |n.u| ds = \int_{\Sigma} (1 - 0) 1 |n.u| ds = 2. \end{aligned}$$

Ces coefficients sont donnés par la matrice $A(1 : 2, 1 : 2)$ calculée par la fonction `exemple1_matrice_elem_Ad_int_LR` ci-dessus avec

$$v(1 : 2, 1 : 2) = \begin{pmatrix} 0 & 0 \\ -2 & 2 \end{pmatrix}$$

4. DESCRIPTION DU CODE F^N `Assemblage_elementaire/matrice_elem_Ad_int_LR`

On considère maintenant l'espace EF $E = P_1^d$ dont les ddl sont représentés à la Figure 4.3-15 et pour $i = 1, \dots, 6$, on note ϕ_i la fonction de base associée au ddl numéro i .

On paramètre l'arête commune aux deux triangles par $t : [0, 1] \rightarrow t(1, 0) + (1 - t)(0, 1)$. On trouve alors, en utilisant ce paramétrage, les relations :

$$\begin{aligned} a_{41} &= \int_{\Sigma} (\phi_1^+ - \phi_1^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} -\phi_1^- \phi_4^+ ds = -2 \int_0^1 s^2 ds = -2/3, \\ a_{42} &= \int_{\Sigma} (\phi_2^+ - \phi_2^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} 0 \phi_4^+ |n.u| ds = 0, \\ a_{43} &= \int_{\Sigma} (\phi_3^+ - \phi_3^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} -\phi_3^- \phi_4^+ ds = -2 \int_0^1 (1-s)s ds = -1/3, \\ a_{44} &= \int_{\Sigma} (\phi_4^+ - \phi_4^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} \phi_4^+ \phi_4^+ ds = -2 \int_0^1 s^2 ds = 2/3, \\ a_{45} &= \int_{\Sigma} (\phi_5^+ - \phi_5^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} \phi_5^+ \phi_4^+ ds = -2 \int_0^1 (1-s)s ds = 1/3, \\ a_{46} &= \int_{\Sigma} (\phi_6^+ - \phi_6^-) \phi_4^+ |n.u| ds = \sqrt{2} \int_{\Sigma} 0 \phi_4^+ |n.u| ds = 0. \end{aligned}$$

Pour les autres termes, par symétrie on trouve :

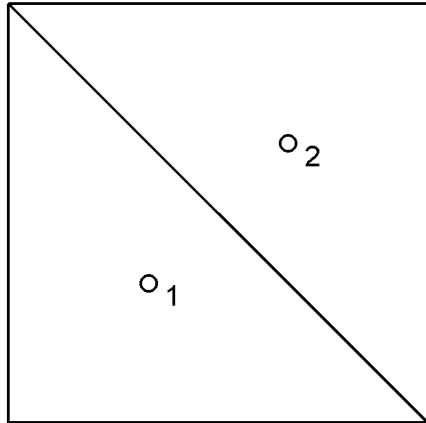
$$\begin{aligned} a_{51} &= a_{43} = -1/3, \quad a_{52} = a_{42} = 0, \quad a_{53} = a_{41} = -2/3, \\ a_{54} &= a_{45} = 1/3, \quad a_{55} = a_{44} = 2/3, \quad a_{56} = a_{46} = 0. \end{aligned}$$

On vérifie ensuite que les autres termes $a_{1i}, a_{2i}, a_{3i}, a_{6i}, 1 \leq i \leq 6$, sont nuls car les fonctions $\phi_1^+, \phi_2^{\pm}, \phi_3^+, \phi_6^{\pm}$ sont nulles sur Σ . Ces coefficients calculés sont donnés par la matrice $A(3 : 8, 3 : 8)$ calculée par la fonction `exemple1_matrice_elem_Ad_int_LR` ci-dessus avec

$$A(3 : 8, 3 : 8) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -2/3 & 0 & -1/3 & 2/3 & 1/3 & 0 \\ -1/3 & 0 & -2/3 & 1/3 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

Numéros des ddl du P_0 discontinu



Numéros des ddl du P_1 discontinu

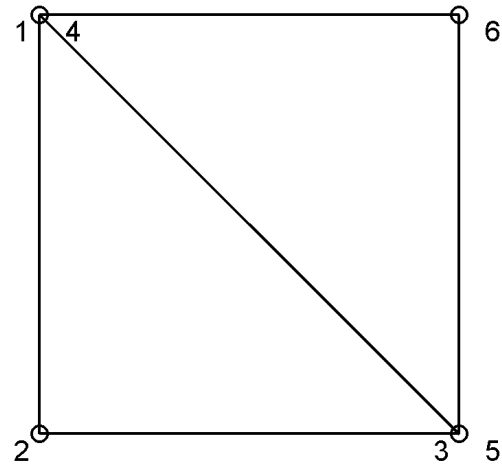


Figure 4.3-15. Représentation des ddl du P_0^d et du P_1^d .

Les calculs ci-dessus sont effectués dans la fonction `matrice_elem_Ad_int_LR` par l'intermédiaire de la fonction exemple `exemple1_matrice_elem_Ad_int_LR`. Cet exemple traite aussi le cas du P_2^d dont les ddl sont représentées à la figure 4.3-16 ci-après. Les termes (a_{ij}) ci-dessus, ainsi que ceux coorepondant au P_2^d sont données par les termes $A(i, j)$. La variable `Ass_Ad` est celle calculée par la fonction `EF_Prep` avec l'utilisation de la fonction `matrice_elem_prepa_Ad`. Nous remettons ci-dessous la numérotation globale ds arêtes du maillage.

Numéros de sommets et d'aretes

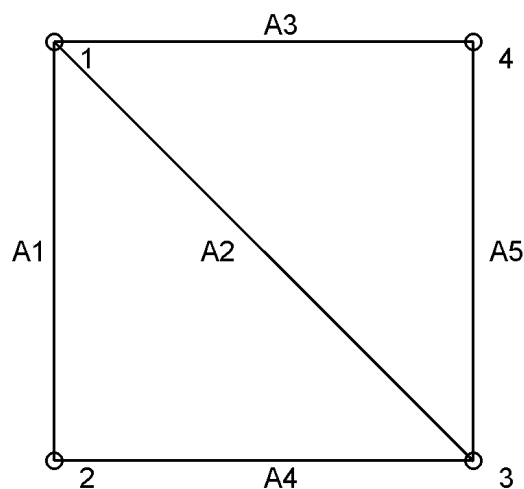


Figure 4.3-13. Numérotation globale des arêtes (figure donnée à la description de la fonction `matrice_elem_Ad`).

4. DESCRIPTION DU CODE F^N `Assemblage_elementaire/matrice_elem_Ad_int_LR`

L'arête intérieure est donc l'arête numéro 2. Ainsi les matrices `Ass_Ad{2, 1, 1, 1}` et `Ass_Ad{2, 1, 1, 2}` donnent les valeurs des fonctions de base locales ϕ_1^2 du P_0^d sur le triangle numéro $\text{Pa}(2, 1) = 2$ *i.e.* K_2 et ϕ_1^1 sur le triangle numéro $\text{Pa}(2, 3) = 1$, *i.e.* K_1 , des formules de quadrature sur les arêtes de ces triangles.

Pour `Ass_Ad{2, 1, 1, 1}` (triangle K_2) on a :

$$\begin{aligned}\phi_1^2(m) &= \text{Ass\_Ad}\{2, 1, 1, 1\}(:, 1) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{1,x}^2(m) &= \text{Ass\_Ad}\{2, 1, 2, 1\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' , \\ \phi_{1,y}^2(m) &= \text{Ass\_Ad}\{2, 1, 3, 1\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' ,\end{aligned}$$

et

$$\begin{aligned}\phi_1^1(m) &= \text{Ass\_Ad}\{2, 1, 1, 2\}(:, 1) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)' , \\ \phi_{1,x}^1(m) &= \text{Ass\_Ad}\{2, 1, 2, 2\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' , \\ \phi_{1,y}^1(m) &= \text{Ass\_Ad}\{2, 1, 3, 2\}(:, 1) = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)' ,\end{aligned}$$

Ensuite les termes relatifs au second espace P_1^d sont donnés par `Ass_Ad{2, 2, k, 1}` et `Ass_Ad{2, 2, k, 2}`, avec $k = 1, 2, 3$ qui figure l'indice de dérivation. On retrouvera ces valeurs à l'exemple 2 de la description de la fonction `matrice_elem_prepa_Ad` qui est mis en œuvre par la fonction `exemple2_matrice_elem_prepa_Ad`. Dans cet exemple l'espace EF produit est l'espace $E = P_1^c \times P_2^c$ et les valeurs correspondantes sont celles relatives au premier espace soit, respectivement `Ass_Ad{2, 1, k, 1}` et `Ass_Ad{2, 1, k, 2}` (comme il s'agit des fonctions de bases locales aux triangles $P_1^c \times P_2^c$ et $P_1^d \times P_2^d$ donnent les mêmes résultats).

Pour les termes relatifs au troisième espace P_2^d on pourra exécuter la fonction `exemple`.

Ensuite, la variable de ttype `CELL Sc_Ad` est celle calculée par la fonction `matrice_elem_Ad_int_LR` par une boucle sur les arêtes du domaine (variable `Pa`). La variable `A` est alors obtenue par concaténation suivante des cellules de `Sc_Ad` avec

```
A = [full(Sc_Ad{1,1}), full(Sc_Ad{1,2}), full(Sc_Ad{1,3}) ; ...
      full(Sc_Ad{2,1}), full(Sc_Ad{2,2}), full(Sc_Ad{2,3}) ; ...
      full(Sc_Ad{3,1}), full(Sc_Ad{3,2}), full(Sc_Ad{3,3})    ] ;
```

Dans cet exemple on pourra modifier la ligne `D.E = [100,101,102]` ; par `D.E = [1,2,3]` pour traiter respectivement le P_1^c , le P_2^c et le P_3^c qui renvoient alors des matrices `Sc_Ad` et `A` nulles.

Dans cette fonction, la syntaxe de la fonction qui définit le champ u suivant lequel on dérive est la suivante :

```
function [Mc, Nc] = eq_Ad(donnees, numF, uc, a1, b1, c1, ...
                          a2, b2, c2, de, Ac, Bc, numK, numAloc, numA) ;
```

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```
Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;
Mc = Un ;
Nc = Mc ;
```

Mc et la valeur aux points de la formule de quadrature de u_1 et Nc celle de u_2 .

Il est à noter que dans la fonction le cas des EF continus est traité en regardant les *ddl communs* aux arêtes et non plus directement sur les numéros des espaces d'EF. Cela permet de tenir compte, par exemple des espaces d'EF discontinus avec des «bulles». L'exemple `exemple3_matrice_elem_Ad_int_LR` permet de considérer ces éléments.

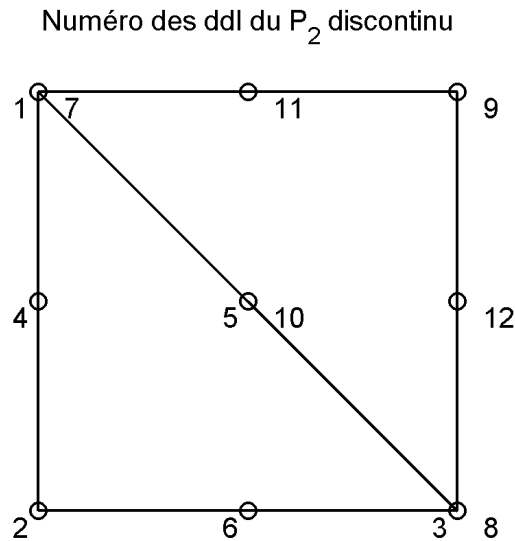


Figure 4.3-16 Représentation des *ddl* du P_2^d .

Exemple 2 : Cet exemple reprend l'exemple précédent avec le fichier

EF/exemples/Assemblage_elementaire/exemple2_matrice_elem_Ad_int_LR.m

en utilisant de manière détaillée, pour illustrer son emploi, la fonction `matrice_elem_prepa_Ad` qui est utilisée à travers `EF_Prep` dans l'exemple 1. La commande est

```
[Ass_Ad, Sc_Ad, A] = exemple2_matrice_elem_Ad_int_LR
```

Exemple 3 : Un exemple de champ de vitesse u qui rentre et qui sort d'une arête intérieure est traité dans la commande

```
cd C :/EF_text/exemples/exemple_transport4, instruct2b
```

Cette commande qui reprend le calcul de la commande

```
cd C :/EF_text/exemples/exemple_transport4, instruct2
```

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

du Paragraphe 2.7.3, mais sur le maillage de la figure 4.3-10 rappelé ci-dessus. Le champ de vitesse u , qui transporte la troisième inconnue, est $u(x, y) = (-y, x)$.

Pour une fonction f ce type de champ introduit une discontinuité des fonctions $f^{\pm, u}$ comme définie en (1.18). Ainsi une formule d'intégration exacte pour un polynôme f ne le sera plus *a priori* pour $f^{\pm, u}$.

Nous donnons maintenant l'algorithme de la fonction `matrice_elem_Ad_int_LR`.

Algorithme :

EF/Assemblage_elementaire/matrice_elem_Ad_int_LR.m

On remet ici la partie déclarative de la fonction :

```
function [Sc_Ad] = matrice_elem_Ad_int_LR(donnees, i, T, Tc, ...
    Pa, X, Y, Uc, D, Mat_syst_Ad_int_LR, It, Ass_Ad, Sc_Ad) ;
```

- Initialisation du compteur des triangles qui contiennent l'arête courante numéro i :

```
nT = 0 ;
```

- On parcourt le premier et l'éventuel second triangle portant l'arête i . On calcule la matrice élémentaire associée à cette arête et à la méthode de Lesaint-Raviart. La matrice élémentaire n'est ajoutée à la matrice globale que si l'arête est intérieure (si elle appartient à 2 triangles) :

```
for at = [0,2]
```

On initialise les variables suivantes :

numK : prend successivement la valeur du numéro du premier triangle auquel appartient l'arête i (cas $at = 0$), puis, si cette arête appartient à un second triangle, la valeur du numéro de ce second triangle et la valeur 0 sinon (cas $at = 2$).

numALoc : si **numK** > 0, cette variable est le numéro local de l'arête i sur le triangle numéro **numK**, elle vaut 0 si **numK** = 0.

numA : numéro global d'arête i ,

numF : type d'arête auquel appartient l'arête : frontière (1,2,3, ...) ou arête intérieure (0).

```
    numK = Pa(i,1 + at) ;
```

```
    numALoc = Pa(i,2 + at) ;
```

```
    numA = i ;
```

```
    numF = Pa(i,5) ;
```

Ci-dessous, si l'arête est sur un triangle (utilisé pour l'éventuel deuxième triangle), on incrémente le compteur de triangle de 1 et on poursuit à l'intérieur du «si».

Ci-dessous **r2** vaut [1,2] si **r** = 1, [1,3] si **r** = 1, [2,3] si **r** = 3, **som(nT, :)** porte les numéros locaux des sommets de l'arête donné par **r2** et **somT(nT)** le numéro du troisième sommet du triangle courant de numéro **numK**. Le terme **n(nT, :)** est le vecteur des composantes de la normale extérieure au triangle :

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```

if numK > 0
    nT = nT + 1 ;
    r = numALoc ;
    r2 = (r-3)*(r-2)*[1,2]/2 + (3-r)*(r-1)*[1,3] + ...
                                                (r-1)*(r-2)*[2,3]/2 ;

    som(nT, :) = T(numK,r2) ;
    somT(nT) = setdiff( T(numK,1 :3), T(numK,r2)) ;
    V1 = [X(som(nT,2)) - X(som(nT,1)), ...
          Y(som(nT,2)) - Y(som(nT,1))] ;
    V2 = [X(somT(nT)) - X(som(1)), ...
          Y(somT(nT)) - Y(som(1))] ;

    %Projection orthogonale
    q = (V1*V2')/(V1*V1') ;
    n(nT, :) = (q*V1 - V2)/norm(V2 - q*V1) ;

```

Initialisation de `ptcB{ic,nT}` le vecteur des numéros des *ddl* de l'espace numéroté `D.E(i)` appartenant au triangle courant :

```

for ic = 1 :length(D.E)
    ptc{ic} = Tc{ic}(numK, :) ;
    ptcB{ic,nT} = ptc{ic} ;
end ;

```

Coordonnées des sommets du triangle courant :

```

XT = X(T(numK,1 :3)) ;
YT = Y(T(numK,1 :3)) ;

```

Valeurs aux *ddl* du triangle des composantes de la fonction *U* représentée par `Uc` :

```

for ic=1 :length(D.f)%Nombre d'espaces de
    %1'espace EF produit
    Uc_pt{ic} = Uc{ic}(ptc{D.f(ic)}) ;
end ;

```

Vecteur `c` indiquant le poids sur l'arête concernée pour les formules de quadrature :

```

c=[0,0,0] ;
c(numALoc)=1 ;

```

Coefficients relatifs à la formule de quadrature sur l'arête `i`, le vecteur `c` permet de mettre à 0 les coefficients pour l'intégration sur les deux autres arêtes du triangle :

```

[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, ...

```

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

```
L1, L2, L3] = assemble_coeff_A(XT, YT, c, D) ;
```

Duplication des coefficients de la formule de quadrature. Le nombre de lignes de $I\{ic\}$ est égal à la dimension de l'espace d'EF de numéro $D.E(ic)$ défini sur le triangle K , ce calcul est effectué pour la mise en œuvre des formules de tyoe (3.5) – (3.9) du Paragraphe 3.1.2 :

```
for ic = 1 :length(D.E)%Nombre de types d'espaces EF
    I{ic} = pre_assemble_A([], D.E(ic), [], a1, b1, c1, ...
        a2, b2, c2, de, Ac, Bc, coeff, 1, D, L1, L2, L3) ;
end ;
```

Valeurs aux points de la formule de quadrature pour l'intégration sur les arêtes du triangle courant de la fonction U (par un exemple un itéré courant) et de ses dérivées, représentée par Uc (formule sur les trois arêtes du triangle), les termes $uc\{ic, jc\}$ sont des vecteurs lignes :

```
for ic = 1 :length(D.f)%Nombre d'espaces de
    %l'espace EF produit
    for jc = 1 :length(D.der)%Derivation : D.der = [0,1,2]
        uc{ic,jc} = pre_assemble_A(Uc_pt{ic}, ...
            D.E(D.f(ic)), D.der(jc), a1, b1, c1, a2, ...
            b2, c2, de, Ac, Bc, coeff, 2, D, L1, L2, L3) ;
    end ;
end ;
```

Valeur des fonctions de base aux points de la formule de quadrature (formule sur les trois arêtes du triangle), $sc\{ic, jc\}$ et $tc\{ic, jc\}$ sont des matrices de type `DOUBLE ARRAY` telles que le nombre de lignes de $sc\{ic, jc\}$ et le nombre de colonnes de $tc\{ic, jc\}$ sont tous deux égal au nombre de points de la formule de quadrature :

```
for ic = 1 :length(D.f)%Nombre d'espaces de
    %l'espace EF produit
    for jc = 1 :length(D.der)%Derivations
        sc{ic,jc} = Ass_Ad{i,D.f(ic),jc,nT} ;
        tc{ic,jc} = sc{ic,jc}' ;
    end ;
end ;
```

Valeurs aux points de la formule de quadrature des composantes du champ de vecteur v suivant lequel on dérive. Ses composantes sont données par les vecteurs lignes Mc , Nc , la fonction qui définit v est indiquée par la variable `Mat_syst_Ad_int_LR` :

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```
[Mc, Nc] = feval(Mat_syst_Ad_int_LR, donnees, numF, uc, ...
    a1, b1, c1, a2, b2, c2, de, Ac, Bc, numK, numALoc, numA);
```

Formule de quadrature sur l'arête considérée. Ci-dessous on calcule à partir de la longueur de `coeff` et pour le triangle courant le vecteur `ind` de longueur `length(coeff)/3` des numéros des points de la formule de quadrature qui interviennent pour l'intégration sur l'arête `i`.

Le construction de `ind` ci-dessous est restreinte aux formules d'intégration du type de celles programmées dans le code où les points d'intégrations sont numérotés dans le sens a_1 vers a_2 , a_1 vers a_3 , a_2 vers a_3 avec a_1 , a_2 , a_3 sommets du triangle courant et avec un nombre égal de points sur chaque arête. Les numéros des points de la formule de quadrature sur l'arête `i` sont alors les suivants :

- si `numALoc` = 1 les numéros vont de 1 à m avec $m = \text{coeff}/3$,
- si `numALoc` = 2 les numéros vont de $m + 1$ à $2m$,
- si `numALoc` = 3 les numéros vont de $2m + 1$ à $3m$.

Les numéros dépendent du numéro local de l'arête et donc du triangle considéré.

La variable `ind` ci-dessous prend la valeurs 1 : m si `numALoc` = 1, $m+1$: $2m$ si `numALoc` = 2 et $2m+1$: $3m$ si `numALoc` = 3, ce qui donne :

```
ind = (numALoc - 1)*length(coeff)/3 + ...
    1 :numALoc*length(coeff)/3 ;
```

On calcule ensuite le vecteur ligne `un(1, :)` qui est la valeur aux points de la formule de quadrature pour l'intégration sur l'arête courante de $v.n$ le produit scalaire du champ v suivant lequel on dérive et de la normale extérieure au premier triangle. Les points de la formule de quadrature sont ceux de la formule de quadrature pour l'intégration sur les arêtes du premier triangle, restreints à ceux utilisés pour l'intégration sur l'arête courante (ci dessous le vecteur `ind` est celui relatif au premier triangle avec la condition du `if` ci-dessous). La normale est la normale extérieure au premier triangle (condition du `if` ci-dessous). Ensuite `un(2, :)` est le vecteur opposée de `un(1, :)` et `unm` prend la valeur de `un(1, :)` :

```
if nT==1%Premier triangle
    un(1, :) = Mc(ind)*n(1,1) + Nc(ind)*n(1,2);
    un(2, :) = -un(1, :);
    unm = un(nT, :);
end;
```

- Traitement du premier triangle.

Valeurs des coefficients d'intégration aux points de la formule de quadrature de l'arête `i` relativement au premier triangle, les coefficients sont dupliques à la dimension de l'espace d'EF numéro `D.E(i)` construit sur un triangle, les valeurs sont mises dans `IA` qui est pour l'intégration sur l'arête `i`, l'analogue de `Ic` :

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

```

if nT==1%Premier triangle
    for ic = 1 :length(D.E)%Nombre de types d'espaces EF
        IA{ic} = I{ic}( :,ind) ;
        ptcA{ic} = ptc{ic} ;
    end ;

```

Dans cette boucle, on calcule sur le premier triangle, ci-dessus IA et ptcA et ci-dessous scA, tcA, scAm, scAp, tcAm, tcAp ces valeurs seront ensuite concaténées avec les valeurs correspondantes sur l'éventuel second triangle

Pour chaque fonction de base f , on calcule les valeurs de f aux point d'intégration de l'arête puis les valeurs de $f^{\pm,v}$ comme défini en (1.18). On notera $f^{\pm} = f^{\pm,v}$. On a $f^+(x) = f \mathbb{I}_{[v.n < 0]} = f \cdot (v.n < 0)$ (vaut f si $v.n < 0$ et 0 si $v.n \geq 0$) $f^-(x) = f \cdot (v.n > 0)$ (vaut f si $v.n > 0$ et 0 si $v.n \leq 0$).

Ces valeurs sont mises dans scA{ic,jc}, scAm{ic,jc}, scAp{ic,jc} qui sont des matrices de type DOUBLE ARRAY de $m = \text{length}(\text{ind})$ lignes et qui représentent respectivement les valeurs de type f , f^- et f^+ des fonctions de base de l'espace EF produit E , *idem* pour tcA{ic,jc}, tcAp{ic,jc}, tcAm{ic,jc} mais qui sont des matrices ayant m colonnes :

```

    for ic = 1 :length(D.f)%Nombre d'espaces de
        %l'espace EF produit
        for jc = 1 :length(D.der)%Derivations
            %Valeur des fonctions de base
            %   restreintes aux points de numero ind
            %   dans la formule de quadrature
            scA{ic,jc} = sc{ic,jc}(ind, :) ;
            tcA{ic,jc} = scA{ic,jc}' ;%
            %Calcul des f+ et f- (points ind)
            sizsc=size(sc{ic,jc}) ;

```

- Ci-dessous le terme $(\text{un}(1, :) < 0)$ est répliqué pour la multiplication :

```

            scAp{ic,jc} = repmat((un(1, :)<0)',1, ...
                                sizsc(2)).*sc{ic,jc}(ind, :) ;
            scAm{ic,jc} = repmat((un(1, :)>0)',1, ...
                                sizsc(2)).*sc{ic,jc}(ind, :) ;
            tcAp{ic,jc} = scAp{ic,jc}' ;
            tcAm{ic,jc} = scAm{ic,jc}' ;
        end ;
    end ;

```


4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```
end;%Fin du if nT==1
```

- Traitement du second triangle.

Ensuite, si $nT = 2$ on traite le deuxième triangle, et on fait une concaténation des résultats précédents. On regarde si $\text{som}(2,:) \neq \text{som}(1,:)$, car alors sur le deuxième triangle, l'arête a un ordre de numérotation inverse de ses sommets par rapport au premier triangle et on tient alors compte du fait que les colonnes de $\text{sc}\{ic, jc\}(\text{ind}, :)$ (resp. les lignes de $\text{tc}\{ic, jc\}(:, \text{ind})$) donnent alors les valeurs des fonctions de base aux points d'intégration énumérés dans le sens inverse du cas précédent. On remarque qu'actuellement dans le code les formules de quadrature sont identiques et symétrique sur chaque arête. Il est cependant nécessaire que cette propriété soit respectée dans la fonction `matrice_elem_Ad_int_LR`, notamment lors de l'utilisation des formules d'intégration (3.17) du Paragraphe 3.1.3 (message d'erreur sinon).

```
if nT==2
```

- Concaténation de `IA` qui contient sur chaque triangle les coefficients des formules de quadrature dupliqués (restrictions aux points de la formule de quadrature de l'arête i) :

```
for ic = 1 :length(D.E)
%Coefficients d'integration restreints a l'arete. On
% duplique la ligne des coefficients d'integration
% relativement a l'arete vue du second triangle
if som(2, :)==som(1, :)
%L'arete i est orientee dans le meme
% sens sur chacun des triangles
IB{ic} = I{ic}( :,ind);
IA{ic} = [IA{ic};IB{ic}];
else
%L'arete i orientee dans un sens inverse
% sur le second triangle
IB{ic} = I{ic}( :,fliplr(ind));
IA{ic} = [IA{ic};IB{ic}];
end;
end;
```

- Concaténation de `ptcA` qui contient les *ddl* de chaque triangle :

```
%Rajout des numeros de DDL du deuxieme triangle
% a ceux du premier
ptcA{ic} = [ptcA{ic}, ptc{ic}];
end;
```

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

- Concaténation de scA , $scAm$, $scAp$, tcA , $tcAm$, $tcAp$ qui sont alors les valeurs des fonctions de base f , f , f^+ de chacun des triangles aux points de la formule de quadrature de l'arête i) :

```
%Valeur des fonctions de base f, f+, f- concatenees
%   a celles du triangle precedent :
for ic = 1 :length(D.f)%Nombre d'espaces de
    %l'espace EF produit
    if som(2, :)==som(1, :)
        %arete i orientee dans le meme
        % sens sur chacun des triangles
        for jc = 1 :length(D.der)%derivation
            scA{ic,jc}=[scA{ic,jc},sc{ic,jc}(ind, :)] ;
            tcA{ic,jc}=scA{ic,jc}' ;%
            %Calcul des valeurs f+ et f-
            sizsc = size(sc{ic,jc}) ;
```

Ci dessous $un(2, :)$ représente la valeur de $v.n$ aux points de la formule de quadrature sur les arêtes du premier triangle restreinte aux points pour l'intégration sur cette arête et n est la normale extérieure au second triangle :

```
    scAp{ic,jc} = [scAp{ic,jc}, ...
        repmat((un(2, :) < 0)', 1, ...
            sizsc(2)).*sc{ic,jc}(ind, :)] ;
    scAm{ic,jc} = [scAm{ic,jc}, ...
        repmat((un(2, :) > 0)', 1, ...
            sizsc(2)).*sc{ic,jc}(ind, :)] ;
    tcAp{ic,jc} = scAp{ic,jc}' ;
    tcAm{ic,jc} = scAm{ic,jc}' ;
end ;
else%arete i orientee en sens inverse
    %   d'un triangle a l'autre
    %Par securite
    if som(2, :) ~= fliplr(som(1, :))
        error('pb aretes communes differentes')
    end ;
    for jc = 1 :length(D.der)%derivation
        scA{ic,jc} = ...
```

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```

[scA{ic,jc},flipud(sc{ic,jc}(ind, :))] ;
tcA{ic,jc} = scA{ic,jc}' ;
%Calcul des valeurs f+ et f-
sizsc = size(sc{ic,jc}) ;

```

Ci-dessous $un(2,:)$ est la valeur de $v.n$ aux points de la formule de quadrature sur les arêtes du premier triangle restreinte aux points pour l'intégration sur cette arête et n est la normale extérieure au second triangle, on garde dans $un(2,:)$ le sens des points sur les arêtes et on l'inverse pour sc :

```

scAp{ic,jc} = ...
[scAp{ic,jc}, repmat((un(2, :) < 0)', ...
1, sizsc(2)).*flipud(sc{ic,jc}(ind, :))] ;
scAm{ic,jc} = ...
[scAm{ic,jc}, repmat((un(2, :) > 0)', ...
1, sizsc(2)).*flipud(sc{ic,jc}(ind, :))] ;
tcAp{ic,jc} = scAp{ic,jc}' ;
tcAm{ic,jc} = scAm{ic,jc}' ;
end;%Fin du for jc = 1 :length(D.der)
end;%Fin du if som(2, :)==som(1, :)
end;%Fin for ic = 1
end;%Fin du if nT==2
end;%Fin du if numK>0
end;%Fin du for at
• Fin du traitement du second triangle, calcul d'un variable annexe :
for ic = 1 :length(D.E)
sizA{ic} = size(IA{ic}) ;
end ;

```

- Prise en compte des espaces d'EF continus et des fonctions bulles. Ce passage est facultatif si on n'utilise que des EF discontinus avec bulles ou non.

Ci-dessous $scAp$, $scAm$ et $tcAp$ sont les valeurs des fonctions de bases locales sur les deux triangles. Dans le cas d'EF continus, les colonnes correspondant à des *ddl* communs à chaque triangle figurent donc en double, elles doivent être ajoutées entre elles (*cf.* fin du Paragraphe 3.5) pour donner la valeur de ces fonctions sur l'arête, puis la redondance de ces fonctions supprimées (les *ddl* figurant en double). On a choisit de supprimer la redondance du côté du second triangle et de mettre le résultat de l'ajout des fonctions de base du côté du premier triangle,

Pour les fonctions de type «bulles», il n'y a pas de nœuds communs mais les fonctions sont nulles au bord en continu ou discontinu, on conserve ces fonctions.

4. DESCRIPTION DU CODE F^N Assemblage_elementaire/matrice_elem_Ad_int_LR

Ci-dessous, à ic fixé (on considère alors le ic -ème type d'espace d'EF), I donne les numéros des ddl en commun (vide dans le cas d'EF discontinus), J est la position de ces ddl communs dans la liste des ddl du premier triangle. Ces ddl communs concernent la première moitié des colonnes de $scAp\{ic,jc\}$, $scAm\{ic,jc\}$ et $tcAp\{ic,jc\}$. Ensuite $dim + K$ est la position des ddl communs dans la liste des ddl du premier triangle suivie de la liste des ddl du second triangle. Ces ddl communs concernent la seconde moitié des colonnes de $scAp\{ic,jc\}$, $scAm\{ic,jc\}$ et $tcAp\{ic,jc\}$.

```

if nT>=2
    for ic=1 :length(D.E)
        dim=length(ptcB{ic,1});%nbr de DLL
        %DDL communs = continuité
        [I,J,K]=intersect(ptcB{ic,1},ptcB{ic,2}) ;
        if ~isempty(J)
            - Modification de IA, sizA et ptcA :
                IA{ic}(end - length(I) + 1 : end, :)=[] ;
                sizA{ic}=sizA{ic} - (length(I)) ;
                %suppression de redondance : on enlève les ddl
                %   redondants (en commun) en enlevant
                %   ceux du côté du second triangle (droite)
                ptcA{ic}(dim + K)=[] ;
            end ;
        end ;
        %restriction à la dérivation d'ordre 0
        jc=1 ;
        for ic=1 :length(D.f)%Nombre d'espaces de
            %l'espace EF produit
            dim=length(ptcB{D.f(ic),1});%nbr de DLL
            %sommets communs = continuité
            [I,J,K] = intersect(ptcB{D.f(ic),1},ptcB{D.f(ic),2}) ;
            if ~isempty(J)
                - Modification de scAp, scAm et tcAp :
                    %Ajout : donne la valeur des fonctions de base
                    %   qui est mise du côté premier triangle (gauche).
                    for n1 = 1 :length(J)

```

4.3. DESCRIPTION DES F^{NSFN} Assemblage_elementaire/matrice_elem_Ad_int_LR

```

        scAp{ic,jc}( :,J(n1)) = ...
        scAp{ic,jc}( :,J(n1)) + scAp{ic,jc}( :,dim+K(n1)) ;
        scAm{ic,jc}( :,J(n1)) = ...
        scAm{ic,jc}( :,J(n1)) + scAm{ic,jc}( :,dim+K(n1)) ;
        tcAp{ic,jc}(J(n1), :) = ...
        tcAp{ic,jc}(J(n1), :) + tcAp{ic,jc}(dim+K(n1), :) ;
    end ;
    %Suppression de redondance
    % (deuxieme triangle, a droite) :
    scAp{ic,jc}( :, dim + K) = [] ;
    scAm{ic,jc}( :, dim + K) = [] ;
    tcAp{ic,jc} = scAp{ic,jc}' ;
end ;
end ;
end ;
• Calcul de la matrice élémentaire avec la formule (3.17) du Paragraphe 3.1.3. Les
espaces d'EF concernés sont désignés par le vecteur It :
for ic = It
    L2c{ic} = IA{D.f(ic)}.* ...
        repmat(abs(unm),sizA{D.f(ic)}(1),1).* ...
        tcAp{ic,1}*(scAp{ic,1} - scAm{ic,1}) ;
end ;
• Ajout à la matrice d'assemblage dans le cas d'une arête intérieure (cf. Paragraphe
3.1.3.1)
if nT >= 2
    for ic = It
        Sc_Ad{ic,ic}(ptcA{D.f(ic)}, ptcA{D.f(ic)}) = ...
            Sc_Ad{ic,ic}(ptcA{D.f(ic)},ptcA{D.f(ic)}) + L2c{ic} ;
    end ;
end ;
end ;

```

Fonction :

```
function L = equationJ(D, J, M, N, s, t, LZ, Mv, Nv) ;
```

Calcul de matrices élémentaires du système.

Cette fonction permet de mettre en œuvre les relations (3.16), (3.17), (3.19), ainsi que (3.16)' et (3.17)' dans le cas de formes bilinéaires sur un espace produit de deux termes distincts, données au Paragraphe 3.1.3.

Soient

M : matrice de type **CELL ARRAY** de taille, notée, $r \times m_2$,

N : matrice de type **CELL ARRAY** de taille $r \times n_2$ telles que $M\{i, j\}$ et $N\{i, j\}$ soient des vecteurs lignes de type **DOUBLE ARRAY** de taille α ,

s : vecteur de type **CELL ARRAY** de taille $m_2 \times 1$ avec $s\{i\}$ matrice de type **DOUBLE ARRAY** de taille $\alpha \times d_{s_i}$,

t : vecteur de type **CELL ARRAY** de taille $n_2 \times 1$ avec $t\{i\}$ matrice de type **DOUBLE ARRAY** de taille notée $d_{t_i} \times \alpha$,

LZ : matrice de type **CELL ARRAY** de taille $n_2 \times m_2$ avec $LZ\{i, j\}$, matrice de type **DOUBLE ARRAY** de taille $d_{t_i} \times d_{s_j}$,

Mv : matrice de type **DOUBLE ARRAY** de taille $r \times m_2$ dont les termes valent 1 sauf si $M\{i, j\}$ est le vecteur nul, on pourra alors poser de manière facultative $Mv(i, j) = 0$ pour accélérer les calculs.

Nv : matrice de type **DOUBLE ARRAY** de taille $r \times n_2$ dont les termes valent 1 sauf si $N\{i, j\}$ est le vecteur nul, on pourra alors poser de manière facultative $Nv(i, j) = 0$ pour accélérer les calculs.

J : vecteur de type **CELL** de taille $1 \times n_2$ avec $J\{i\}$ matrice de type **DOUBLE ARRAY** de taille $d_{t_i} \times \alpha$,

D : facultatif, peut être mis à [] (cette variable n'intervient pas dans la version actuelle de **equationJ**),

alors la commande

```
L = equationJ(D, J, M, N, s, t, LZ, Mv, Nv) ;
```

calcule **L** la matrice de type **CELL ARRAY** de taille $n_2 \times m_2$, telles que $L\{i, j\}$ est une matrice de type **DOUBLE ARRAY** de taille $d_{t_i} \times d_{s_j}$ définie par :

$L\{i, j\} = \dots$

$$LZ\{i, j\} + (J\{i\} \cdot t\{i\}) \cdot (\text{ones}(\text{size}(t\{i\}, 1), 1) * Z\{i, j\}) * s\{j\} \quad (4.1)$$

ou de manière équivalente, par

$L\{i, j\} = \dots$

$$L\{i, j\} + (J\{i\} \cdot t\{i\}) \cdot (Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :)) * s\{j\}; \quad (4.1)'$$

comme programmé dans la fonction `equationJ`.

Remarque. Les parenthèses qui encadrent $J\{i\} * t\{i\}$ sont facultatives

Ci-dessus Z a la valeur :

$$Z\{i, j\} = \sum_{k=1}^r N\{k, i\} * M\{k, j\}, 1 \leq i \leq n_2, 1 \leq j \leq m_2. \quad (4.2)$$

Détaillons la valeur prise par Z . Les matrices M et N représentent des matrices par blocs dont les termes (i, j) sont des vecteurs lignes. Soient $M(p)$ et $N(p)$ les matrices de terme (i, j) donné par $M_{ij}(p) = M\{i, j\}(p)$ et $N_{ij}(p) = N\{i, j\}(p)$, avec $1 \leq p \leq \alpha$, alors $Z\{i, j\}(p)$ est le produit matriciel suivant :

$$Z\{i, j\}(p) = (N'(p)M(p))_{ij}.$$

Décrivons, maintenant le terme L , la taille des termes intervenant dans la formule donnant sa valeur, ainsi qu'une expression de $L\{i, j\}\{p, q\}$.

Le terme $Z\{i, j\}$ est un vecteur ligne de type `DOUBLE` de taille $1 \times \alpha$, le terme

$$Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :)$$

est alors une matrice de taille $d_{t_i} \times \alpha$ avec $d_{t_i} = \text{size}(t\{i\}, 1)$ qui réplique le terme $Z\{i, j\}$ et dont chaque ligne est égale au vecteur $Z\{i, j\}$. Le terme

$$(J\{i\} * t\{i\}) * (Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :))$$

est aussi une matrice de taille $d_{t_i} \times \alpha$. Enfin

$$(J\{i\} * t\{i\}) * (Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :)) * s\{j\}$$

est donc une matrice de taille de taille $d_{t_i} \times d_{s_j}$.

Le terme $L\{i, j\}(p, q)$ est alors le produit de la ligne p de

$$(J\{i\} * t\{i\}) * (Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :))$$

qui est aussi la ligne p de

$$J\{i\}(1, p) * t\{i\}(1, p) * Z\{i, j\}$$

et de la colonne q de $s\{j\}$. On en déduit que

$$\begin{aligned} L\{i, j\}(p, q) &= J\{i\}(p, :) * t\{i\}(p, :) * Z\{i, j\} * s\{j\}(:, q) \\ &= \text{sum}(J\{i\}(p, :) * t\{i\}(p, :) * Z\{i, j\} * s\{j\}(:, q)') \\ &= \text{sum}(J\{i\}(p, :) * Z\{i, j\} * s\{j\}(:, q)' * t\{i\}(p, :)). \end{aligned} \quad (4.3)$$

La fonction `equationJ` permet par exemple le calcul de la matrice A des relations (3.17) et (3.18) du Paragraphe 3.1.3. Rappelons ci-dessous ces relations, on a (en changeant n_p en d_p dans les formules) :

$$A_{pq} = (C^{d_p} \cdot * \Omega_{pq}^{d_p} \cdot * f_{,k'}^p) \cdot (f_{,k}^q)' \quad 1 \leq p, q \leq n. \quad [(3.17)]$$

avec

$$\Omega_{pq}^d = \sum_{k=1}^t N_{kp}^d \cdot * M_{kq}^d, \quad 1 \leq p, q \leq n, \quad [(3.18)]$$

Dans ces relations, A est la matrice de la forme bilinéaire définie par la donnée de l'espace $E(K)$ et des deux fonctions à valeurs matricielles M et N du Paragraphe 3.1.3, on a posé $d_p = \dim(E_p(K))$ au lieu de n_p car ici n_2 désigne le nombre de colonnes de N .

On retrouve la formule calculée par `equationJ` de la manière suivante. On pose $m_2 = n_2 = n = \dim(E(K))$, $r = t$. En modifiant les lettres des indices, les 2 relations précédentes s'écrivent de manière équivalente :

$$A_{ij} = (C^{d_i} \cdot * \Omega_{ij}^{d_i} \cdot * f_{,k'}^i) \cdot (f_{,k}^j)',$$

avec

$$\Omega_{ij}^{d_i} = \sum_{k=1}^r N_{ki}^{d_i} \cdot * M_{kj}^{d_i}, \quad 1 \leq i, j \leq n,$$

qui est la formule calculée par `equationJ` en posant

$$\begin{aligned} J\{\mathbf{i}\} &= C^{d_i}, \quad M\{\mathbf{i}, \mathbf{j}\} = M_{ij}^1, \quad N\{\mathbf{i}, \mathbf{j}\} = N_{ij}^1, \\ \mathbf{s}\{\mathbf{j}\} &= f_{,k}^j, \quad \mathbf{t}\{\mathbf{i}\} = f_{,k'}^i, \quad Z\{i, j\} = \Omega_{ij}^1. \end{aligned}$$

La matrice $f_{,k'}^i$ étant de taille $d_i \times \alpha$, ce choix de $\mathbf{t}\{\mathbf{i}\}$ donne alors $d_{t_i} = d_i$ et des dimensions entre J et $\mathbf{t}$ qui sont donc bien compatibles.

De la même manière, on retrouve l'égalité (4.3) vue ci-dessus :

$$L\{\mathbf{i}, \mathbf{j}\}(p, q) = \text{sum}(J\{\mathbf{i}\}(p, :) \cdot * Z\{\mathbf{i}, \mathbf{j}\} \cdot * \mathbf{s}\{\mathbf{j}\}(:, q)' \cdot * \mathbf{t}\{\mathbf{i}\}(p, :)),$$

en partant de la formule d'intégration appliquée à la relation vue en (3.16) Paragraphe 3.1.3 :

$$\begin{aligned} (A_{ij})_{pq} &= \int_K (M \phi_{q,k}^j e_j, N \phi_{p,k'}^i e_i) = \int_K (N^t M)_{ij} \phi_{q,k}^j \phi_{p,k'}^i \\ &\simeq \sum_{a=1}^{\alpha} c_a \omega_{ij}(m_a) \phi_{q,k}^j(m_a) \phi_{p,k'}^i(m_a). \end{aligned} \quad [(3.16)]$$

Le cas de l'assemblage d'une forme bilinéaire sur deux espaces différents, de la forme $E(K) \times F(K)$, se traite en posant

$$\mathbf{t}\{\mathbf{i}\} = g_{,k'}^i$$

dans la relation (3.17)' :

$$A_{ij} = (C^{m_i} \cdot * \Omega_{ij}^{m_i} \cdot * g_{,k'}^i) * (f_{,k}^j)', 1 \leq i \leq m, 1 \leq j \leq n. \quad [(3.17)']$$

avec ci-dessus $m_i = \dim(F_i(K)) = d_{t_i}$, Ω étant donné par

$$\Omega_{ij}^{m_i} = \sum_{k=1}^t N_{ki}^{m_i} \cdot * M_{kj}^{m_i}, 1 \leq i \leq m, 1 \leq j \leq n. \quad [(3.18)']$$

avec $r = t$. Ce cas est détaillé ci-après avec aussi le cas des formes linéaires.

Les variables $\mathbf{Mv}$ et $\mathbf{Nv}$ servent à accélérer les calculs en restreignant le calcul de $\mathbf{L}$ ci-dessus aux indices (i, j) tels que le terme (i, j) de $\mathbf{Nv}' \star \mathbf{Mv}$ est non nul.

Dans le code EF la fonction `equationJ` permet de calculer les deux membres du système de la façon suivante.

Assemblage du premier membre.

On considère deux espaces d'EF sur un triangle K :

$$E(K) = E_1(K) \times \dots \times E_{m_2}(K),$$

$$F(K) = F_1(K) \times \dots \times F_{n_2}(K),$$

(par rapport aux notations du Paragraphe 3.1.3 on a la correspondance $n = m_2$ et $m = n_2$), M une fonction de K à valeurs dans l'espace des matrices de taille $r \times m_2$, avec r donné définissant le produit scalaire. Soit N une fonction K à valeur dans l'espace des matrices de taille $r \times n_2$. On note

$$(s_i^1 e_1)_{1 \leq i \leq \dim(E_1(K))}, \dots, (s_i^{m_2} e_{m_2})_{1 \leq i \leq \dim(E_{m_2}(K))}$$

la base choisie pour E et

$$(t_i^1 e_1)_{1 \leq i \leq \dim(F_1(K))}, \dots, (t_i^{n_2} e_{n_2})_{1 \leq i \leq \dim(F_{n_2}(K))}$$

la base choisie pour F avec s_i^q, t_i^q fonctions de K dans $\mathbb{R}$ et $(e_i)_{1 \leq i \leq k}$ base canonique de $\mathbb{R}^k$, $k \in \mathbb{N}$.

La fonction `equationJ` permet alors, à l'erreur d'intégration près, de calculer la matrice élémentaire d'assemblage par blocs élémentaires

$$(\Omega_{pq})_{ij} = \int_K (M s_{j,D1}^q e_q, N t_{i,D2}^p e_p),$$

avec $1 \leq i \leq \dim(F_p), 1 \leq j \leq \dim(E_q), 1 \leq p \leq n_2, 1 \leq q \leq m_2$ et $D1 = 0, 1, 2, D2 = 0, 1, 2$ désignant une dérivation, par la commande

$L = \text{equationJ}(D, J, M, N, s, t, LZ, Mv, Nv) ;$

où $M\{i, j\}$ est le vecteur ligne des valeurs du terme M_{ij} aux points de la formule de quadrature, *idem* pour $N\{i, j\}$.

Ensuite $s\{i\}$ est la matrice dont la colonne j contient les valeurs aux points de la formule de quadrature de $s_{j,D1}^i$, s_j^i étant la j -ème fonction de la base de $E_i(K)$ et $t\{i\}$ la matrice dont la ligne j contient les valeurs de $t_{j,D2}^i$ avec t_j^i la j -ème fonction de base de $F_i(K)$.

Enfin, $J\{i\}$ représente la matrice dont chaque ligne représente le vecteur ligne des coefficients de la formule d'intégration répliqué $\dim(F_i(K))$ fois.

Le calcul est fait en appliquant les formules (3.16)', (3.17)' et (3.18)' du Paragraphe 3.1.3. On remarque que dans le code on aura $E(K) = F(K)$ et que les formules (3.16), (3.17) et (3.18) s'appliquent aussi.

Assemblage du second membre.

On conserve les notations du cas précédent. On note

$$(f_i^1 e_1)_{1 \leq i \leq d_1}, \dots, (f_i^{m_2} e_{m_2})_{1 \leq i \leq d_{m_2}}$$

un ensemble de fonctions de K à valeurs dans $\mathbb{R}^{m_2}$, avec f_i^q fonctions de K dans $\mathbb{R}$. En pratique dans le code on aura

$$d_1 = d_2 = \dots = d_{m_2} = 1$$

et les fonctions correspondent successivement aux composantes successives du second membre :

$$f = (f_1^1, \dots, f_1^{m_2}) = \sum_{q=1}^{m_2} f_1^q e_q.$$

La fonction `equationJ` permet alors de calculer la matrice élémentaire d'assemblage par blocs élémentaires :

$$(B_{pq})_{ij} = \int_K (M f_j^q e_q, N t_{i,D2}^p e_p),$$

avec $1 \leq i \leq \dim(F_p), 1 \leq j \leq d_q, 1 \leq p \leq n_2, 1 \leq q \leq m_2$ et $D2 = 0, 1, 2$ dérivation, par

$L = \text{equationJ}(D, J, M, N, s, t, LZ, Mv, Nv) ;$

Ci-dessus $s\{i\}$ est la matrice dont la colonne j contient les valeurs aux points de la formule de quadrature de la j -ème fonction de $(f_j^i e_i)_{1 \leq j \leq d_i}$ i.e. f_j^i .

Dans le cas $d_1 = d_2 \cdots = d_{m_2} = 1$ on aura :

$$(B_{pq})_{i1} = \int_K (M f_1^q e_q, N t_i^p e_p), \quad 1 \leq i \leq \dim(F_p), 1 \leq p \leq n_2, 1 \leq q \leq m_2,$$

(B_{pq}) est alors un vecteur colonne de $\dim(F_p(K))$ termes qui correspond à la valeur de B de (3.16)'' ou, de manière identique, à (3.17)''. Soit b la forme linéaire définie sur $F(K)$ par

$$b(v) = \int_K (M f, N v, D_2)$$

et, comme au Paragraphe 3.1.3.2, F le vecteur second membre associé à cette forme linéaire

$$\begin{aligned} (F_p)_i &= \sum_{q=1}^{m_2} \int_K (M f_1^q e_q, N t_{i,D_2}^p e_p) \\ &= \sum_{q=1}^{m_2} \int_K (N^t M')_{pq} f_1^q t_{i,D_2}^p, \quad 1 \leq i \leq \dim(F_p), \quad 1 \leq p \leq n_2. \end{aligned}$$

On obtient donc F à partir de la valeur de B en sommant sur q les termes

$$(B_{pq})_{i1} = \int_K (M f_1^q e_q, N t_{i,D_2}^p e_p)$$

qui redonne bien le résultat :

$$\sum_{q=1}^{m_2} (B_{pq})_{i1} = \sum_{q=1}^{m_2} \int_K (M f_1^q e_q, N t_{i,D_2}^p e_p) = (F_p)_i, \quad 1 \leq i \leq \dim(F_p), \quad 1 \leq p \leq n_2.$$

La sommation se fait en dehors de la fonction `equationJ`, par exemple, pour l'assemblage sur les triangles, cette sommation est effectuée dans la fonction `matrice_elem.T_scm` avec une boucle sur le nombre de second membre qui est choisi à 3 dans le code (*cf.* (1.11)), pour tenir compte des 3 types de dérivations 0, 1 et 2 considérées dans le code.

Variables d'entrée :

D : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à `donnees.D`. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe 2.2.1. Variable facultative, la valeur de cette variable n'intervient pas dans la version actuelle de la fonction. Elle peut être mise ici à [].

J : CELL ARRAY. Vecteur ligne. Les cellules $J\{i\}$ sont des matrices de type DOUBLE ARRAY ayant toutes le même nombre de colonne noté **Q**.

Dans le code, le nombre de termes de **J** est égal à n le nombre d'espaces d'EF composant l'espace EF produit $E = F$. Le nombre **Q** est le nombre de points de la

formule de quadrature utilisée et $J\{i\}$ est la matrice de dimension $\dim(E_i(K)) \times Q$ dont chaque ligne est donnée par les coefficients de la formule de quadrature choisie, $E_i(K)$ étant l'espace d'EF sur un triangle K correspondant au i -ème espace de E . Cette formule de quadrature est indiquée par `donnees.D.intT` (triangles) ou `donnees.D.intA` (arêtes).

M : CELL ARRAY. **M**{i, j} DOUBLE ARRAY : vecteur ligne de Q termes.

N : CELL ARRAY. **N**{i, j} DOUBLE ARRAY : vecteur ligne de Q termes. Le nombre de lignes de **N** doit être égal au nombre de lignes de **M** (ce qui correspond à la compatibilité du produit du matriciel $N'M$) : `size(N, 1) = size(M, 1)` et le nombre colonnes au nombre de termes de **J** : `size(N, 2) = length(J)`.

Dans le code le nombre de colonnes de **N** est égal à n (nombre d'inconnue). Le nombre de colonnes de **M** est aussi égal à n dans le cas de la forme bilinéaire définissant le premier membre et peut être quelconque (pour une page donnée) pour la forme linéaire définissant le second membre (*cf.* les tailles de ces matrices aux Eqns. (1.6), (1.8), (1.11), (1.12)).

s : CELL ARRAY. Vecteur colonne de `size(M, 2)` termes. **s**{i} DOUBLE ARRAY : matrice de Q lignes, pas de conditions sur le nombre de colonnes.

t : CELL ARRAY. Vecteur colonne de `size(N, 2)` termes. **t**{i} DOUBLE ARRAY : matrice de Q colonnes et de même nombre de lignes que la matrice $J\{i\}$.

Dans le code, pour l'assemblage de la forme linéaire, on a `size(s{i}, 2) = 1` et `size(t{i}, 1) = dim(E_i(K))` et pour l'assemblage de la forme bilinéaire, on a la relation `size(s{i}, 2) = size(t{i}, 1) = dim(E_i(K))`.

Mv : CELL DOUBLE ARRAY de même taille que **M**. **Mv**(i, j) est nul si **M**{i, j} est un vecteur nul et vaut 1 sinon.

Nv : CELL DOUBLE ARRAY de même taille que **N**. **Nv**(i, j) est nul si **N**{i, j} est un vecteur nul et vaut 1 sinon.

LZ : CELL ARRAY de taille `size(N, 2) × size(M, 2)`. **LZ**{p, q} DOUBLE ARRAY : matrice de taille `size(t{p}, 1) × size(s{q}, 2)`.

La commande

```
cd C :/EF/exemples/Assemblage_elementaire,
    exemple1b_matrice_elem_T_scm
```

donne un exemple de calcul de second membre où le nombre de ligne et de colonnes de **M** peut différer du nombre d'inconnues (suivant la «page» définissant le second membre) et la commande

```
cd C :/EF_text/exemples/exemple_stokes, instruct1
```

pour la résolution d'un écoulement de Poiseuille au Paragraphe 2.10.4 donne un exemple où le nombre de lignes de **M** diffère du nombre d'inconnue dans le calcul du premier membre.

Variables de sortie :

L : CELL ARRAY. L Matrice de même taille que LZ, les matrices L{i, j} et LZ{i, j} sont aussi de même taille. La valeur de L est donnée par :

$$L\{i, j\} = LZ\{i, j\} + (J\{i\} \cdot t\{i\}) \cdot (Z\{i, j\}(\text{ones}(1, \text{size}(t\{i\}, 1)), :)) \cdot s\{j\};$$

Exemples :

Exemple 1. Cet exemple est donné par le fichier de commandes

EF/exemples/Assemblage_elementaire/exemple_equationJ.m

Dans cet exemple les variables d'entrée de la fonction **equationJ** sont les suivantes : on pose $D = []$.

La matrice **M** est de taille (comme matrice de type **CELL**) 4×5 et la matrice **N** de taille 4×3 pour donner la matrice **L** de taille 3×5 .

On a choisi pour la valeur des cellules de **M** et de **N** une valeur constante qui est le vecteur

$$v = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10].$$

On a donc :

$$M = \begin{Bmatrix} v, & v, & v, & v, & v; \\ v, & v, & v, & v, & v; \\ v, & v, & v, & v, & v; \\ v, & v, & v, & v, & v; \end{Bmatrix} \quad N = \begin{Bmatrix} v, & v, & v; \\ v, & v, & v; \\ v, & v, & v; \end{Bmatrix}$$

avec ce choix la matrice **Z** définie par

$$Z\{i, j\} = \sum_{k=1}^r N\{k, i\} \cdot M\{k, j\}, 1 \leq i \leq n_2, 1 \leq j \leq m_2.$$

vaut

$$Z = \begin{Bmatrix} 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v; \\ 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v; \\ 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v, & 4v.*v; \end{Bmatrix}$$

Soit **u** le vecteur :

$$u = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1].$$

Le vecteur de **s** est choisi égal à

$$s = \begin{Bmatrix} [u', & u', & u']; \\ [u', & u', & u']; \\ [u', & u', & u', & u', & u', & u']; \\ [u', & u', & u', & u', & u', & u']; \\ [u', & u']; \end{Bmatrix}$$

et le vecteur $\mathbf{t}$:

$$\mathbf{t} = \begin{Bmatrix} [u; & u; & u; & u; & u]; \\ [u, & u, & u; & u; & u]; \\ [u; & u; & u; & u; & u; & u; & u; & u]. \end{Bmatrix}$$

Les termes (i, j) matrice LZ qui doit être de taille 3×5 sont initialisés à la matrice nulle de taille

$$\mathbf{LZ}\{i, j\} = \mathbf{zeros}(\mathbf{size}(\mathbf{t}\{i\}, 1), \mathbf{size}(\mathbf{s}\{j\}, 2)).$$

Les matrices $\mathbf{Mv}$ et $\mathbf{Nv}$ doivent valoir nécessairement :

$$\mathbf{Mv} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \mathbf{Nv} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix},$$

puisque les cellules de $\mathbf{M}$ et $\mathbf{N}$ sont toutes non nulles (sinon c'est une valeur différente de $\mathbf{L}$ dans (4.1) qui sera calculée).

La relation (4.3) vue plus haut

$$\mathbf{L}\{i, j\}(p, q) = \mathbf{sum}(\mathbf{J}\{i\}(p, :) * \mathbf{Z}\{i, j\} * \mathbf{s}\{j\}(:, q)' * \mathbf{t}\{i\}(p, :)),$$

donne alors

$$\mathbf{L}\{i, j\}(p, q) = \mathbf{sum}(u * (4 * v.^2) * u * u) = 1540.$$

Dans cet exemple on calcule aussi $\mathbf{Z}$, donné par $\mathbf{omega}$ ci-dessous et $\mathbf{L}$ avec les relations (4.2), (4.3) :

EF/exemples/Assemblage_elementaire/exemple_equationJ.m (extrait)

```
%Calcul detaillé
%a) Calcul de omega :
for p=1 :alpha
    for i=1 :size(N,1)
        for j=1 :size(N,2)
            Np(i,j) = N{i,j}(p) ;
        end ;
    end ;
    for i=1 :size(M,1)
        for j=1 :size(M,2),
            Mp(i,j) = M{i,j}(p) ;
```

```

        end ;
    end ;
    for i=1 :size(N,2)
        for j=1 :size(M,2)
            P = Np'*Mp ;
            omega{i,j}(p) = P(i,j) ;
        end ;
    end ;
end ;

%b) calcul de L par deux formules differentes :
for i=1 :size(N,2)
    for j=1 :size(M,2),
        for p=1 :size(t{i},1)
            for q=1 :size(s{j},2)
                L2{i,j}(p,q) = ...
                    (J{i}(p, :).*t{i}(p, :)).*omega{i,j}*s{j}( :,q) ;
                L3{i,j}(p,q) = ...
                    sum(J{i}(p, :).*omega{i,j}*s{j}( :,q)'.*t{i}(p, :)) ;
            end ;
        end ;
    end ;
end ;

```

Exemple 2. Nous considérons l'exemple de l'équation

$$N'M \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} u \\ v \\ w \end{pmatrix}_{,x} = f(x, y)$$

sur le triangle K de sommets $(0, 0)$, $(1, 0)$, $(0, 1)$ avec

$$M = N = \begin{pmatrix} x & 0 & 0 \\ y & x+y & y \\ x & x & x \end{pmatrix}, f(x, y) = \begin{pmatrix} 2x^4 + 2x^2y^2 + 2y^3x + y^4 + x^3y + 2x \\ 2x^3y + 4x^2y^2 + x^4 + 3y^3x + y^4 \\ 2x^2y^2 + x^4 + 2y^3x + y^4 + x^3y + y \end{pmatrix};$$

et qui possède la solution

$$u(x, y) = x^2, v(x, y) = y^2, w(x, y) = xy.$$

A titre d'exemple on résout ce problème sur l'espace EF produit :

$$E = P_4(K) \times P_5(K) \times P_4(K).$$

La résolution est faite dans le fichier

EF/exemples/Assemblage_elementaire/exemple3_equationJ.m

La formulation variationnelle utilisée est celle obtenue en multipliant, avec le produit scalaire sur $\mathbb{R}^3$, les équations par les fonctions tests de l'espace produit et en intégrant sur le triangle.

Dans ce fichier, on initialise d'abord la variable D (qui est la valeur du champ de `donnees.D`) qui décrit l'espace EF produit E . Puis on calcule les formules de quadrature sur le triangle K avec la fonction `assemble_coeff_T`. Les valeurs des fonctions de base et de leur dérivées en x sont calculées par la fonction `pre_assemble_T` avec les variables `s` et `sx` pour les valeurs des fonctions de base mises en colonnes et `t` pour les valeurs mises en ligne.

Soient `x` et `y` les vecteurs lignes des coordonnées cartésiennes des points de la formule de quadrature. Soient `Zero` = $(0, \dots, 0)$, `Un` = $(1, \dots, 1)$ les vecteurs de même longueur que `x` et `y` alors M et N prennent la valeur

$$M = N = \begin{Bmatrix} x & \text{Zero} & \text{Zero} \\ y & x + y & y \\ x & x & x \end{Bmatrix}.$$

Les matrices associées à la dérivation sont données par

$$M10 = N10 = \begin{Bmatrix} \text{Un}, & \text{Zero}, & \text{Zero}; \\ \text{Zero}, & \text{Un}, & \text{Zero}, \\ \text{Zero}, & \text{Zero}, & \text{Un} \end{Bmatrix},$$

et le second membre, où ci-dessous `x` et `y` sont cette fois-ci les vecteurs colonnes des coordonnées cartésiennes des points de la formule de quadrature :

$$\begin{aligned} f = & \{ 2.*x.^4 + 2.*x.^2.*y.^2 + 2.*y.^3.*x + y.^4 + x.^3.*y + 2.*x ; \\ & 2.*x.^3.*y + 4.*x.^2.*y.^2 + x.^4 + 3.*y.^3.*x + y.^4 ; \\ & 2.*x.^2.*y.^2 + x.^4 + 2.*y.^3.*x + y.^4 + x.^3.*y + y \} ; \end{aligned}$$

avec comme matrices associées :

$$M = N = \begin{Bmatrix} \text{Un}, & \text{Zero}, & \text{Zero}; \\ \text{Zero}, & \text{Un}, & \text{Zero}, \\ \text{Zero}, & \text{Zero}, & \text{Un} \end{Bmatrix}.$$

On vérifie que l'on retrouve la solution en comparant la solution calculée et la solution exacte aux *ddl* de l'espace EF E .

On remarque que par rapport à la représentation utilisée pour les seconds membres dans le code, le contenu de chaque cellules de **f** est ici un vecteur colonne au lieu d'être un vecteur ligne (*cf.* par exemple le Paragraphe 1.3 pour la représentation des seconds membres). Dans le code, les vecteurs lignes contenus dans les cellules des seconds membres sont transformés en vecteur colonnes dans l'algorithme des fonctions `matrice_elem_T_scm`, `matrice_elem_A_scm`, `matrice_elem_Ad_scm` avant leur traitement par la fonction `equationJ` qui intervient dans ces trois fonctions.

Le fichier

EF/exemples/Assemblage_elementaire/exemple2_equationJ.m

reprend le même problème, moins le terme de dérivation, avec :

$$N'M \begin{pmatrix} u \\ v \\ w \end{pmatrix} = f(x, y),$$

avec les seconds membres donnés par la solution,

$$u(x, y) = x^2, v(x, y) = y^2, w(x, y) = xy$$

comme la solution ci-dessus, ainsi que par

$$u(x, y) = 1, v(x, y) = 1, w(x, y) = 1.$$

Le *script* du fichier *exemple3_equationJ.m* est le suivant :

EF/exemples/Assemblage_elementaire/exemple3_equationJ.m

```
%exemple3 equationJ. Utilisation de la fonction d'assemblage
%   EF/Assemblage_elementaire/equationJ
%
%   On resout deux problemes sur un domaine maille par un seul
%   triangle K. L'espace EF E est le P4(K) x P5(K) x P4(K). Les
%   solutions exactes sont choisies dans cette espace.
%   Resolution de
%
%       N'M(u,v,w) + (u,v,w),x = f = (f1, f2, f3) ;
%
%   avec
%
%   f = (2x^4 + 2x^2y^2 + 2y^3x + y^4 + x^3y + 2x ;
```

4. DESCRIPTION DU CODE

FN Assemblage_elementaire/equationJ

```
%      2x^3y + 4x^2y^2 + x^4 + 3y^3x + y^4 ;
%      2x^2y^2 + x^4 + 2y^3x + y^4 + x^3y + y} ;
%
%  et M et N donnees par :
%
%  M = N = [x, 0, 0 ; y, x + y, y ; x, x ,x] .
%
%  Le probleme a pour solution :
%
%      (u,v,w) = (x^2, y^2, x*y) .
%
%  La formulation variationnelle est celle obtenue en
%  multipliant avec le produit scalaire de R^3 les
%  equations par les fonctions tests de l'espace EF
%  produit et en integrant ensuite sur K.

clear

%Variables de maillage :
T = [1,2,3] ;
X = [0,1,0] ;
Y = [0,0,1] ;

%Variables decrivant les espaces d'EF, les types
%types d'espaces d'EF sont  E1 = P4(K) et E2 = P5(K) :
D.E = [4,5] ;

%Derivation (toujours fixe dans ce code), facultatif ici :
D.der = [0,1,2] ;

%Numero de la formule de quadrature sur les triangles :
D.intT = 10 ;

%Numero de la formule de quadrature sur les aretes
```

```

% facultatif ici :
D.intA = 4;

%Choix de l'espace EF produit : ici E = E1 x E2 x E1 :
D.f = [1,2,1];

%Dimensions locales des espaces EF composant l'espace produit,
% sert uniquement pour initialier LZ, s, t ci-dessous :
dim = [15,21,15];

%Coefficients des formules de quadrature :
[A1, B1, C1, A2, B2, C2, de, Ac, Bc, coeff, L1, L2, L3] = ...
                                assemble_coeff_T(X, Y, D);

%Calcul des fonctions de bases (valeurs des derivees d'ordre 0) :
der1 = 0;
der1x = 1;
der2 = 0;
for i=1 :length(D.f)
    %s{i} : Valeurs des fonctions de base par colonnes
    % (chiffre 3 en entree de pre_assemble_T) :
    s{i,1} = pre_assemble_T([], D.E(D.f(i)), der1, A1, B1, ...
        C1, A2, B2, C2, de, [], [], coeff, 3, D, L1, L2, L3);
    %sx{i} : Valeurs des derivees par rapport a x des fonctions de
    % base par colonnes (chiffre 3 en entree de pre_assemble_T) :
    sx{i,1} = pre_assemble_T([], D.E(D.f(i)), der1x, A1, B1, ...
        C1, A2, B2, C2, de, [], [], coeff, 3, D, L1, L2, L3);
    %t{i} : Valeurs des fonctions de base par lignes
    % (chiffre 4 en entree de pre_assemble_T) :
    t{i,1} = pre_assemble_T([], D.E(D.f(i)), der2, A1, B1, ...
        C1, A2, B2, C2, de, [], [], coeff, 4, D, L1, L2, L3);
end;

%Vecteur ligne unite et nul de taille Ac :
```

```

Un = ones(1,length(Ac)) ;
Zero = zeros(1,length(Ac)) ;

%Coordonnees des points de la formule de quadrature :
x = Ac ;
y = Bc ;

%Initialisation des autres arguments de la fonction EQUATIONJ pour
% le calcul de la matrice du systeme.
%a) Coefficients de la formules de quadrature pour chaque espace
% EF :
J{1} = repmat(coeff,dim(1),1) ;
J{2} = repmat(coeff,dim(2),1) ;
J{3} = repmat(coeff,dim(3),1) ;

%b1) Matrices definissant la forme bilineaire, la derivation
% concerne est der1 = der2 = 0 qui vient de s et t ci-apres
% dans EQUATIONJ :
M = {x, Zero, Zero ;
      y, x + y, y ;
      x, x, x} ;
N = M ;
[m1,m2] = size(M) ;
[n1,n2] = size(N) ;

%b2) Matrices definissant la forme bilineaire, la derivation
% concerne est der1x = 0 qui vient de sx ci-apres
% dans EQUATIONJ :
M10 = {Un,Zero,Zero ;
        Zero, Un, Zero ;
        Zero, Zero, Un} ;
N10 = M10 ;

%c) Initialisation de LZ qui est de la taille de la variable

```

```

% de sortie de EQUATIONJ :
for i=1 :n2
    for j=1 :m2
        LZ{i,j} = zeros(dim(i), dim(j)) ;
    end ;
end ;

%d) Matrices definissant pour acclereler les calculs, les
% coefficients sont ici tous egaux a 1 :
Mv = ones(size(M)) ;
Nv = ones(size(N)) ;

%Calcul de la matrice d'assemblage du systeme, on peut mettre
% D = [] ci-dessous :
L = equationJ(D, J, M, N, s, t, LZ, Mv, Nv) ;

%Calcul de la matrice d'assemblage du systeme :
L = equationJ([], J, M10, N10, sx, t, L, Mv, Nv) ;

%a) Matrices intervenant dans la definition de la forme lineaire,
% la derivation concerne est der2 = 0 qui vient de t ci-apres
% dans EQUATIONJ :
M = {Un, Zero, Zero ;
      Zero, Un, Zero ;
      Zero, Zero, Un} ;
N = M ;

%b) Definition du second membre f definissant la forme lineaire
x = Ac' ;
y = Bc' ;
%On part de (u,v,w) = (x^2, y^2, x*y) et on calcule le second
% membre N'M(u,v,w) = f donne par les valeurs ci-dessous :
f = {2.*x.^4 + 2.*x.^2.*y.^2 + 2.*y.^3.*x + y.^4 + x.^3.*y + 2.*x ;
      2.*x.^3.*y + 4.*x.^2.*y.^2 + x.^4 + 3.*y.^3.*x + y.^4 ;

```

```

2.*x.^2.*y.^2 + x.^4 + 2.*y.^3.*x + y.^4 + x.^3.*y + y};

%c) Initialisation de LZ pour la forme lineaire, les cellules (i,j)
%   de LZ sont de taille size(t{i},1) x size(f{i},2)
%   = size(t{i},1) x 1 (f joue le role de s dans le cas du premier
%   membre) :
clear LZ
for i=1 :n2
    for j=1 :m2
        LZ{i,j} = zeros(dim(i),1);
    end;
end;

%Matrice d'assemblage du second membre :
F = equationJ(D, J, M, N, f, t, LZ, Mv, Nv);

%Matrice du systeme :
A = [ L{1,1},L{1,2},L{1,3};
      L{2,1},L{2,2},L{2,3};
      L{3,1},L{3,2},L{3,3} ];

%Second membre du systeme par sommation des contributions de
%   chaque composante du second membre multipliee par les
%   fonctions tests successives de chaque espace de l'espace EF
%   produit multipliees avec le coefficient N'M correspondant.
%   Ici seul F[i,i] est non nul :
B = [ sum([F{1,1},F{1,2},F{1,3}],2);
      sum([F{2,1},F{2,2},F{2,3}],2);
      sum([F{3,1},F{3,2},F{3,3}],2) ];

%Resolution du systeme :
C = A\B;

%Erreur solution exacte - solution approchee aux DDL.

```

```
%a) Initialisation de la variable decrivant les donnees :
donnees.D = D ;

%b) Calcul des DDL :
[Xc, Yc, Tc, Ed, Arc] = ddl(donnees,X,Y,T,[]) ;

%c) Calcul de l'erreur aux DDL :
disp(['Erreur solution exacte - solution approchee aux DDL', ...
      ' en norme du max (1.7093e-011) :']) ;
disp(max(abs(C - [Xc{1}]'.^2 ; Yc{2}]'.^2 ; Xc{1}]'.*Yc{1}]')) ;
```

Algorithme :

C :/EF/Assemblage_elementaire/equationJ.m

On remet ici la partie déclarative de la fonction :

```
function L = equationJ(D, J, M, N, s, t, LZ, Mv, Nv) ;
```

- Initialisation de L pour la sommation

```
L = LZ ;
```

Ensuite ci-dessous la sous-fonction `mult_cp` associe aux variables M, N, Mv, Nv les variables I1, J1 sont deux vecteurs colonnes de même taille, telles que

$$(I1(m), J1(m))_{1 \leq m \leq \text{length}(I)}$$

décrit les indices (i, j) de la matrice $Nv' * Mv$ pour lesquels $(Nv' * Mv)_{ij}$ est non nul. Le terme $Z\{i, j\}$ prend alors la valeur suivante qui correspond au produit de deux matrices dont les coefficients sont des vecteurs (*cf.* (4.2)) :

$$Z\{i, j\} = \sum_{k=1}^r N\{k, i\} \cdot M\{k, j\}, 1 \leq i \leq n_2, 1 \leq j \leq m_2, \text{ si } (Nv' * Mv)_{ij} \neq 0,$$

avec $r = \text{size}(M, 1)$, $m_2 = \text{size}(M, 2)$, $n_2 = \text{size}(M, 2)$ et

$$Z\{i, j\} = [] \text{ si } (Nv' * Mv)_{ij} = 0.$$

%On calcule pour les indices tels que $L(i, j)$ est non nulle

```
[Z, I1, J1] = mult_cp(M, N, Mv, Nv) ;
```

On fait ensuite la boucle pour $1 \leq m \leq \text{length}(I)$ pour le calcul des termes $L\{I1m, J1m\}$ avec la formule (4.1) :

```

for m=1 :length(I1) ;
    I1m=I1(m) ;
    J1m=J1(m) ;
    L{I1m,J1m} = L{I1m,J1m} + ...
    (J{I1m}.*t{I1m}).*(Z{I1m,J1m}(ones(1,size(t{I1m},1)),:))*s{J1m} ;
end ;

```

Lorsque $(Nv' * Mv)_{ij} = 0$, la somme

$$\sum_{k=1}^r N\{k, i\} * M\{k, j\}, 1 \leq i \leq n_2, 1 \leq j \leq m_2,$$

est donc le terme

$$(J\{j\} * t\{i\}) * (Z\{i, j\}(ones(1, size(t\{i\}, 1)), :)) * s\{j\};$$

est le vecteur nul. Donc dans la boucle ci-dessus ajoute bien toutes les contributions non nulles à L.

- Fin de la fonction

Voici ci-dessous la sous-fonction `mult_cp` utilisée dans `equationJ`. Cette fonction permet d'accélérer de manière significative les calculs pour les problèmes EF vus dans ce document car les matrices M et N sont relativement creuses.

```

function [Z,I,J] = mult_cp(M,N,Mv,Nv) ;
[m1,m2] = size(M) ;
Z = [] ;
[I,J] = find(Nv'*Mv) ;

```

Les calculs sont fait dans des boucles décroissantes pour l'allocation mémoire :

```

for m=length(I) :-1 :1

```

- Calcul de $Z\{I(m), J(m)\}$

```

    Im = I(m) ;
    Jm = J(m) ;
    S = [] ;
    for k=m1 :-1 :1 ;
        S(k, :) = N{k, Im}.*M{k, Jm} ;
    end ;

```

- Vectorisation de la somme :

```

    Z{Im, Jm} = sum(S, 1) ;
end ;

```


Fonction :

```
function L = assemble_equ(D, I, Mcc, Ncc, sc, tc) ;
```

Assemblage et sommation de matrices élémentaires de système.

Soit J définie par $J\{i\} = I\{D.f(i)\}$ pour $i = 1$ à $\text{length}(D.f)$.

Cas 1. Si Mcc et Ncc sont des tableaux de type **CELL** de taille 1×3 alors

```
L = assemble_equ(D, I, Mcc, Ncc, sc, tc) ;
```

effectue dans une boucle pour $1 \leq d1, d2 \leq 3$ l'opération

```
L = equationJ(D, J, Mcc{d1}, Ncc{d2}, sc(:,d1), ...
              tc(:,d2), L, Mv{d1}, Nv{d2}) ;
```

Cas 2. Si Mcc et Ncc sont des tableaux de type **CELL** de taille 3×3 c'est l'opération

```
L = equationJ(D, J, Mcc{d1,d2}, Ncc{d1,d2}, ...
              sc(:,d1), tc(:, d2), L, Mv{d1,d2}, Nv{d1,d2}) ;
```

qui est effectuée. On remarque qu'une sommation sur L est effectuée dans les deux cas dans `equationJ`.

Ci-dessus D est de type **STRUCT ARRAY** et a un champ $D.f$ vecteur de type **DOUBLE ARRAY**, I est un vecteur ligne de type **CELL**, sc et tc sont de type **CELL ARRAY** de 3 colonnes, Mv , Nv sont des tableaux de type **CELL** de même taille que, respectivement, Mcc et Ncc . Ces variables sont telles que J comme définie ci-dessus, M et N définies par $M = Mcc\{d1\}$, $N = Ncc\{d2\}$ dans le cas 1 et par $M = Mcc\{d1, d2\}$, $N = Ncc\{d1, d2\}$ dans le cas 2, $s = sc(:, d1)$, $t = tc(:, d2)$, $LZ = L$, Mw et Nw définies par $Mw = Mv\{d1\}$, $Nw = Nv\{d2\}$ dans le cas 1 et par $Mw = Mv\{d1, d2\}$, $Nw = Nv\{d1, d2\}$ dans le cas 2 respectent les conditions sur les variables d'entrée pour

$$L = \text{equationJ}(D, J, M, N, s, t, LZ, Mw, Nw);$$

qui s'écrivent :

$\text{size}(N, 1) = \text{size}(M, 1)$ (compatibilité produit matriciel $N' \times M$),

$\text{size}(M, 2) = \text{size}(s, 1)$ (compatibilité produit matriciel $M \times s$),

$\text{size}(N, 2) = \text{size}(t, 1) = \text{length}(J)$ (compatibilité avec le produit matriciel $N \times t$ et aussi compatibilité avec le nombre d'inconnues dans le cas des fonctions tests),

$\text{size}(t\{k\}, 1) = \text{size}(J\{k\}, 1)$, $k = 1$ à $\text{length}(J)$ (compatibilité dimension produit $J\{k\} \cdot t\{k\}$).

D'autre part, comme la taille de L ci-dessus est alors $\text{size}(N, 2) \times \text{size}(M, 2)$ et que le contenu de $L\{i, j\}$ est de taille $\text{size}(t\{i\}, 1) \times \text{size}(s\{j\}, 2)$ (voir la fonction `equationJ`) il faut donc que $\text{size}(N, 2)$, $\text{size}(M, 2)$, $\text{size}(t, 1)$, $\text{size}(s, 1)$ soient des constantes qui ne dépendent pas de $d1$ et $d2$, de même que $\text{size}(t\{i\}, 1)$ et $\text{size}(s\{i\}, 2)$.

Dans le code ces conditions sont toujours vérifiées car :

- dans le cas du calcul de la matrice du système on a $\text{size}(N, 2) = \text{size}(M, 2) = \text{size}(t, 1) = \text{size}(s, 1) = n$, où n est le nombre d'espaces composant l'espace EF produit E de la forme

$$E = E_1 \times \dots \times E_n$$

et $\text{size}(t\{i\}, 1) = \text{size}(s\{i\}, 2) = \dim(E_i(K))$, $E_i(K)$ étant l'espace d'EF sur un triangle K correspondant au i -ème espace de E ,

- dans le cas du calcul du second membre on a $\text{size}(M, 2) = \text{size}(s, 1) = n'$ nombre de composantes du second membre en prenant les seconds membres de taille constante (cf. (1.11), (1.12) à D fixé) et $\text{size}(s\{i\}, 2) = 1$.

Le nombre de ligne de M , qui est égal au nombre de lignes de N , peut lui varier en fonction de **d1** et **d2**.

Remarque. En entrée de **assemble_equ** on a donc $\text{size}(N, 2) = \text{length}(D.f)$.

Notons α un entier quelconque qui dans le code sera le nombre de points de la formule de quadrature.

En entrée de la fonction **assemble_equ**, **D** est une variable de type **STRUCT ARRAY** utilisée par son champ **D.f**, vecteur ligne de **DOUBLE ARRAY**.

I est un vecteur ligne de type **CELL ARRAY**, tel que $I\{k\}$ est une matrice de type **DOUBLE ARRAY** dont chaque ligne contient α termes.

Ensuite, $Mcc\{i, j\}$ et $Ncc\{i, j\}$ sont des tableaux de cellules, tels que $Mcc\{i, j\}\{p, q\}$ et $Ncc\{i, j\}\{p, q\}$ sont des vecteurs lignes de **DOUBLE** de α termes.

Ensuite **sc** et **tc** sont des tableaux de type **CELL** de 3 colonnes, tels que **sc** $\{p, q\}$ est une matrice de **DOUBLE** dont chaque colonne est un vecteur de α composantes, et tel que **tc** $\{p, q\}$ est une matrice de **DOUBLE** dont chaque ligne est un vecteur de α composantes.

Enfin, **Mv** $\{i, j\}$ est une matrice de double tel que le terme (p, q) vaut 0 si tous les termes de $M\{i, j\}\{p, q\}$ sont nuls et vaut 1 sinon.

Idem pour **Nv** mais relativement à **Ncc**.

En sortie **L** est un tableau de type **CELL** de taille $\text{size}(t, 1) \times \text{size}(s, 1) = \text{size}(N, 2) \times \text{size}(M, 2)$.

Les conditions de compatibilité sont détaillées dans la description des variables ci-dessous et dans la rubrique «conditions sur les dimensions des variables» qui suit.

Dans le code, la taille de **L** correspond à la taille de la matrice par blocs A de taille $n \times n$ de (3.2) du Paragraphe 3.1 ou bien à la matrice par blocs B de taille $n \times n'$ de (3.2)'' du Paragraphe 3.1 (lorsque $E = F$ dans ce Paragraphe).

La valeur de **L** calculée par **assemble_equ** est

- soit la valeur obtenue en calculant pour chaque valeur de dérivation k et k' avec $0 \leq k, k' \leq 2$ la matrice A , Eqns. (3.16), (3.17) du Paragraphe 3.1.3 et en sommant sur k et k' les matrices obtenues,

- soit en calculant le second membre B , matrice par blocs de taille $n' \times n$, Eqns. (3.16)'' (ou de manière identique (3.17)') du Paragraphe 3.1.3.2 et en sommant sur la «dérivation» k' , les 3 seconds membres possibles (cf. (1.11), (1.12)).

Il est à noter que pour obtenir le second membre du système à partir de B , il faut effectuer la sommation sur le deuxième indice de B comme décrit dans (3.19), Paragraphe 3.1.3.2 (cf. aussi (3.3), Paragraphe 3.1).

Variables d'entrée :

Ci-dessous α désigne le nombre de points de la formule de quadrature.

D : STRUCT ARRAY. Variable de description des espaces d'EF correspondant à **donnees.D**. Cette variable représente le champ **D** du Tableau 2.2.1-1c, Paragraphe 2.2.1. La fonction **assemble_equ** utilise uniquement le champ **D.f** qui décrit l'espace produit EF.

I : CELL ARRAY. Vecteur ligne.

$I\{k\}$ = DOUBLE ARRAY. $I\{k\}$ est une matrice de α colonnes. Dans le code $I\{k\}$ représente les coefficients de la formule de quadrature de l'espace d'EF de numéro **D.E(k)** noté $F(K)$, dupliqués sur $\dim(F(K))$ lignes.

Mcc : CELL ARRAY. Tableau 3×3 ou 1×3 de cellules. Relativement aux problèmes EF 3 ci-dessous correspond au nombre de dérivation : $u_{,0}$, $u_{,1}$, $u_{,2}$ numérotées 1, 2 et 3.

$Mcc\{i, j\}$: CELL ARRAY. Tableau de cellules de taille $m_{1ij} \times m_{2ij}$.

Compatibilité : le nombre de colonnes de $Mcc\{i, j\}$ doit être constant et vaut $\text{size}(Mcc\{i, j\}, 2) = \text{size}(L, 2)$, où **L** est la variable de sortie.

$Mcc\{i, j\}\{p, q\}$: DOUBLE ARRAY : Vecteur ligne de α termes.

Ncc : CELL ARRAY. Tableau 3×3 ou 1×3 de cellules de même taille que **Mcc**.

$Ncc\{i, j\}$: CELL ARRAY. Tableau de cellules de taille $n_{1ij} \times n_{2ij}$.

Compatibilité : le nombre de colonnes de $Ncc\{i, j\}$ doit être constant et vaut $\text{size}(Ncc\{i, j\}, 2) = \text{size}(L, 1)$.

$Ncc\{i, j\}\{p, q\}$: DOUBLE ARRAY : Vecteur ligne de α termes.

Compatibilité : On doit avoir $m_{1ij} = n_{1ij}$ (cf. $N' \times M$ ci-dessus).

sc : CELL ARRAY de 3 colonnes.

Compatibilité. Le nombre de terme de chaque colonne $sc(:, q)$ vérifie (cf. $M \times s$ ci-dessus) :

$$\text{size}(sc(:, q), 1) = \text{size}(L, 2) = \text{size}(Mcc\{i, j\}, 2), \forall 1 \leq q \leq 3.$$

$sc\{p, q\}$: DOUBLE ARRAY : matrice de α lignes et de n_p colonnes, sa taille ne dépend pas de q (voir ci-dessus L est de taille $\text{size}(t\{i\}, 1) \times \text{size}(s\{j\}, 2)$, donc, avec $s = sc(:, q)$ $\text{size}(s\{j\}, 2) = \text{size}(sc\{j, q\}, 2)$ doit rester constant pour tout q à j fixé).

Compatibilité : A p fixé on a :

$$n'_p = \text{size}(\text{sc}\{p, q\}, 2) = \text{size}(\text{sc}\{p, 1\}, 2) = \text{constante}$$

et

$$n'_p = \text{size}(\text{L}\{i, p\}, 2),$$

pour tout $1 \leq i \leq \text{size}(\text{L}, 1)$.

Relativement aux espaces d'EF le nombre de colonne n'_p est soit la dimension de $E_p(K)$ soit vaut 1 dans le cas du calcul du second membre, et q désigne soit les trois dérivations soit les trois seconds membres dans (1.11), (1.12).

tc : CELL ARRAY de 3 colonnes.

Compatibilité. Le nombre de termes de chaque colonne **tc**(:, q) vérifie (cf. $N \times t$ ci-dessus) :

$$\text{size}(\text{tc}(:, q), 1) = \text{size}(\text{L}, 1) = \text{size}(\text{Ncc}\{i, j\}, 2), \forall 1 \leq q \leq 3.$$

tc{ p, q } : DOUBLE ARRAY : matrice de α colonnes et de n_p lignes (ne dépend pas de q).

Compatibilité : A p fixé on a :

$$n_p = \text{size}(\text{tc}\{p, q\}, 1) = \text{size}(\text{tc}\{p, 1\}, 1) = \text{constante}.$$

et

$$n_p = \text{size}(\text{L}\{p, i\}, 1),$$

pour tout $1 \leq i \leq \text{size}(\text{L}, 2)$, où **L** est la variable de sortie.

Compatibilité : On doit aussi avoir

$$\text{size}(\text{tc}(:, q), 1) = \text{length}(\text{J}) = \text{length}(\text{D.f})$$

$$n_p = \text{size}(\text{J}\{p\}, 1) \text{ avec } \text{J}\{p\} = I\{D.f(p)\}.$$

Dans le code, relativement aux espaces d'EF, n_p est la dimension de $E_p(K)$ et $q = 1, 2, 3$ désigne les trois dérivations.

Mv : CELL ARRAY de mêmes dimensions que **Mcc**.

Mv{ i, j } : DOUBLE ARRAY. **Mv**{ i, j } est une matrice de terme (p, q) valant 0 si tous les termes de $M\{i, j\}\{p, q\}$ sont nuls et 1 sinon.

Nv : CELL ARRAY de mêmes dimensions que **Ncc**. Définition identique à **Mv**, mais relativement à **Ncc**.

Remarque. Dans le Tableau 2.2.4-1 seconde partie, le second membres **Fc**, qui représente la variable de sortie des fonctions utilisateurs qui définissent le second membre, est un vecteur de type CELL qui a la structure suivante. Soit $Z = \text{Fc}(:, :, k)$ la page k de **Fc** alors Z est de taille 1×3 , $Z\{1, i\}$ représente trois seconds membres

F_1, F_3, F_3 . La cellule $Z\{1, i\}$ est alors un vecteur colonne de n' termes qui représente les n' composantes de chaque second membre, et chacun de ces termes est un vecteur ligne de taille α . En transposant dans Z chacun de ces vecteurs et en posant $Z = [Z\{1, 1\}, Z\{1, 2\}, Z\{1, 3\}]$ on obtient une représentation des 3 seconds membres F_1, F_3, F_3 analogue à celle de **sc**. C'est cette valeur de Z qui est alors transmise à la fonction **assemble_equ** par les fonctions **matrice_elem_T_scm**, **matrice_elem_A_scm**, **matrice_elem_Ad_scm**.

Variables de sortie :

L : CELL ARRAY. Matrice de taille $\text{size}(\text{Ncc}\{1, 1\}, 2) \times \text{size}(\text{Mcc}\{1, 1\}, 2)$, qui vaut aussi $\text{size}(\text{tc}(:, 1), 1) \times \text{size}(\text{sc}(:, 1), 1)$ (cf. taille de L ci-dessus).

$L\{p, q\}$: DOUBLE ARRAY. Matrice de taille : $\text{size}(\text{tc}\{p, 1\}, 1) \times \text{size}(\text{sc}\{q, 1\}, 2)$ de DOUBLE (cf. taille de $L\{p, q\}$ ci-dessus).

Dns le code, pour le premier membre on a $n'_p = n_p = \dim(E_p(K))$ et $L\{p, q\}$ est de taille $n_p \times n_q$ ensuite, pour le second membre on a $n'_p = 1$ et $n_p = \dim(E_p(K))$ et $L\{p, q\}$ est de taille $n_p \times 1$.

Conditions sur les dimensions des variables. On détaille ci-dessous d'où viennent les conditions de compatibilités ci-dessus.

Ce sont les conditions qui proviennent de la fonction **equationJ** et sur la compatibilité de la variable **L** calculée par **equationJ**.

On pose pour $k = 1$ à $\text{length}(\text{D.f})$

$$J\{k\} = I\{D.f(k)\}$$

Soit $1 \leq d1, d2 \leq 3$. Posons M et N telles que si $\text{size}(\text{Mcc}) = [1, 3]$ alors

$$M = \text{Mcc}\{d1\}, N = \text{Ncc}\{d2\}$$

et si $\text{size}(\text{Mcc}) = [3, 3]$ alors

$$M = \text{Mcc}\{d1, d2\}, N = \text{Ncc}\{d1, d2\}.$$

Posons ensuite

$$s = \text{sc}(:, d1), t = \text{tc}(:, d2).$$

Les conditions sur les dimensions sont celles de l'appel de

$$L = \text{equationJ}(\cdot, J, M, N, s, t, \cdot, \cdot, \cdot).$$

et celles dues au fait que les tailles de L et du contenu de ses cellules doivent être indépendants de $d1$ et $d2$.

Pour les variables d'entrée de **equationJ** les conditions sur les dimensions sont :

- a) $\text{size}(N, 1) = \text{size}(M, 1)$ (compatibilité produit matriciel $N'M$),
- b) $\text{size}(M, 2) = \text{size}(s, 1)$ (compatibilité produit matriciel $M \times s$),
- c) $\text{size}(N, 2) = \text{size}(t, 1) = \text{length}(J) = \text{length}(D.f)$ (compatibilité du produit matriciel $N \times t$ et compatibilité avec le nombre d'inconnues dans le cas des fonctions tests),
- d) $\text{size}(t\{k\}, 1) = \text{size}(J\{k\}, 1)$, $k = 1$ à $\text{length}(J)$ (compatibilité dimension du produit $J\{k\} \cdot t\{k\}$).

En sortie L a pour dimension $\text{size}(N, 2) \times \text{size}(M, 2)$ et ses cellules (i, j) la dimension $\text{size}(t\{i\}, 1) \times \text{size}(s\{j\}, 2)$.

Ainsi la compatibilité pour les tailles de L et de ses cellules donne comme conditions :

- b') $\text{size}(M, 2) = \text{size}(s, 1) = C^{ste}$ indépendante de $d1$ et $d2$ (correspond au nombre d'inconnues ou au nombre de composantes du second membre).
- c') $\text{size}(N, 2) = \text{size}(t, 1) = C^{ste}$ indépendante de $d1$ et $d2$. Cette condition est déjà vérifiée dans c) ci-dessus avec la constante valant $\text{length}(D.f)$ (correspond au nombre d'inconnues dans le code).

et à i fixé :

- d') $\text{size}(t\{i\}, 1) = C^{ste}$ indépendante de $d1$ et $d2$. Cette condition est déjà vérifiée dans d) ci-dessus avec la constante valant $J\{i\} \cdot t\{i\}$.
- e) $\text{size}(s\{i\}, 2) = C^{ste}$ indépendante de $d1$ et $d2$.

d') et e) correspondent à la dimension des espaces d'EF avec $\text{size}(t\{i\}, 1) = \dim(E_i(K))$ et $\text{size}(s\{i\}, 2) = \dim(E_i(K))$ dans le cas du système ou bien $\text{size}(s\{i\}, 2) = 1$ dans le cas du second membre.

Les conditions sur L sont toujours vérifiées dans le code en prenant, dans le cas des second membres, des seconds membres ayant tous le même nombre de composantes (cf. (1.11), (1.12), à D fixé).

En conclusion les conditions se résument aux conditions a), b'), c), d) et e). Dans le code les conditions utilisateurs à vérifier portent essentiellement sur la compatibilité des produit matrices vecteurs de type Ms (inconnue), Nt (fonctions test), Mf (second membre) et de la compatibilité du produit matriciel $N'M$.

Relativement aux problèmes EF, la matrice L représente la matrice par blocs de l'assemblage élémentaire du système donnée en (3.16), Paragraphe 3.1.3 (cf. aussi (3.2) au Paragraphe 3.1) ou avant leur sommation pour obtenir le second membre donné en (3.19), les termes en B donné par (3.17)'' au Paragraphe 3.1.3.2 (cf. aussi (3.3), Paragraphe 3.1).

Exemple 1. Cet exemple est donné par le fichier

EF/examples/Assemblage_elementaire/exemple1b_assemble_equ.m.

Le calcul de la matrice L ci-dessus se fait avec la commande

```
cd C :/EF/exemples/Assemblage_elementaire, ...
```

```
L = exemple1_assemble_equ ;
```

qui renvoie la valeur de L sous forme de la variable L tableau de type **CELL**, qui est de taille 2×2 . La valeur de la matrice S est alors la suivante. Soit K le triangle de sommets $a = (0, 0)$, $b = (1, 0)$, $c = (0, 1)$, on note λ_i les coordonnées barycentriques relatives à ces trois sommets.

On calcule L la matrice d'assemblage sur K du système associé à la forme bilinéaire sur $[P_1(K)]^2 \times [P_1(K)]^2$:

$$a(u, v) = \int_K u_1 v_1 + u_{1,x} v_{1,x} + u_{1,y} v_{1,y} + u_2 v_2 + u_{1,x} v_{1,x} + u_{2,y} v_{2,y},$$

en le mettant sous la forme

$$a(u, v) = \int_K \sum_{0 \leq i, j \leq 2} (M_{ij} u_i, N_{ij} v_j)$$

avec les vecteurs $u = (u_1, u_2)'$, $v = (v_1, v_2)'$ et les matrices

$$M_{00} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; M_{11} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}; M_{22} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix};$$

$$M_{ij} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \text{ si } 0 \leq i \neq j \leq 2, \quad N = M.$$

On vérifie que la matrice L du système est donnée par

$$L = \begin{pmatrix} \tilde{L} & 0 \\ 0 & \tilde{L} \end{pmatrix}$$

où $\tilde{L}$ est la somme des trois contributions

$$\tilde{L} = L_{0ij} + L_{1ij} + L_{2ij} = \frac{1}{24} \begin{pmatrix} 26 & -11 & -11 \\ -11 & 14 & 1 \\ -11 & 1 & 14 \end{pmatrix},$$

où les termes L_{kij} , pour $1 \leq i, j \leq 3$ et $k = 0, 1, 2$ sont définis par :

$$L_{kij} = \int_K \lambda_{j,k} \lambda_{i,k}.$$

Pour cela on utilise la formule d'intégration

$$\int_K \lambda_1^i \lambda_2^j = \int_K \lambda_1^i \lambda_3^j = \int_K \lambda_2^i \lambda_3^j = 2|K| \frac{i!j!}{(2+i+j)!},$$

qui donne

$$\int_K \lambda_k^2 = 1/12, 1 \leq k \leq 3, \int_K \lambda_k \lambda_m = 1/24, 1 \leq k \neq m \leq 3.$$

Ensuite les valeurs des dérivées des coordonnées barycentriques données au Paragraphe 3.4, valeurs que l'on peut aussi obtenir en remarquant que pour des polynômes de degré 1 on a ici $\lambda_{i,x} = \lambda_i(a) - \lambda_i(b)$, $\lambda_{i,y} = \lambda_i(a) - \lambda_i(c)$, sont les suivantes :

$$\lambda_{1,x} = -1, \lambda_{1,y} = -1, \lambda_{2,x} = 1, \lambda_{2,y} = 0, \lambda_{3,x} = 0, \lambda_{3,y} = 1.$$

On en déduit que

$$\begin{aligned} L_{0ij} &= \int_K \lambda_j \lambda_i = \begin{pmatrix} 1/12 & 1/24 & 1/24 \\ 1/24 & 1/12 & 1/24 \\ 1/24 & 1/24 & 1/12 \end{pmatrix} \\ L_{1ij} &= \int_K \lambda_{j,x} \lambda_{i,x} = \begin{pmatrix} 1/2 & -1/2 & 0 \\ -1/2 & 1/2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ L_{2ij} &= \int_K \lambda_{j,y} \lambda_{i,y} = \begin{pmatrix} 1/2 & 0 & -1/2 \\ 0 & 0 & 0 \\ -1/2 & 0 & 1/2 \end{pmatrix}. \end{aligned}$$

On obtient finalement pour $\tilde{L}$:

$$\tilde{L} = L_{0ij} + L_{1ij} + L_{2ij} = \frac{1}{24} \begin{pmatrix} 26 & -11 & -11 \\ -11 & 14 & 1 \\ -11 & 1 & 14 \end{pmatrix}.$$

Exemple 2. Cet exemple, qui traite le cas de l'assemblage d'une forme linéaire, est donné par le fichier

EF/exemples/Assemblage_elementaire/exemple1b_assemble_equ.m.

Il est effectué par la commande

```
cd C :/EF/exemples/Assemblage_elementaire, ...
```

```
L = exemple1b_assemble_equ ;
```

qui renvoie la valeur de L sous forme d'un vecteur donnant la valeur du vecteur F décrit ci-après.

On reprend les notations précédente. Dans cet exemple on calcule le second membre F associé à la forme linéaire sur $[P_1(K)]^2$:

$$b(v) = \int_K f_{10}v_1 + f_{11}v_{1,x} + f_{22}v_{1,y} + f_{20}v_2 + f_{21}v_{1,x} + f_{22}v_{2,y},$$

qui s'écrit

$$b(v) = \int_K \sum_{0 \leq i, j \leq 2} (M_{ij} f_i, N_{ij} v_{,j})$$

avec le vecteur $v = (v_1, v_2)'$ et les matrices M et N ci-dessus et f_i donnée par :

$$f_1 = (f_{11}, f_{12})', f_2 = (f_{21}, f_{22})', f_3 = (f_{31}, f_{32})'.$$

On effectue le calcul avec

$$f_1(x, y) = f_2(x, y) = f_3(x, y) = (x^2, y^2) = (\lambda_2^2, \lambda_3^2).$$

Le second membre est alors donné par $F = (F_1, F_2)$ avec

$$(F_1)_j = \int_K \lambda_2^2 \lambda_j + \lambda_2^2 \lambda_{j,x} + \lambda_2^2 \lambda_{j,y} = \begin{pmatrix} 1/60 - 1/12 - 1/12 \\ 1/20 + 1/12 \\ 1/60 + 1/12 \end{pmatrix} = \frac{1}{60} \begin{pmatrix} -9 \\ 8 \\ 6 \end{pmatrix}$$

et

$$(F_2)_j = \int_K \lambda_3^2 \lambda_j + \lambda_3^2 \lambda_{j,x} + \lambda_3^2 \lambda_{j,y} = \begin{pmatrix} 1/60 - 1/12 - 1/12 \\ 1/60 + 1/12 \\ 1/20 + 1/12 \end{pmatrix} = \frac{1}{60} \begin{pmatrix} -9 \\ 6 \\ 8 \end{pmatrix}.$$

On remarque que dans la fonction exemple, la sommation dans (3.19) des termes B_{pq} donnés par (3.17)'' (cf. Paragraphe 3.1.3.2) est effectuée en dehors de la fonction `assemble_equ` avec

```
L = assemble_equ(D, I, Mcc, Ncc, fc, tc) ;
```

```
L = [L{1,1} + L{1,2} ; L{2,1} + L{2,2}] ;
```

Ci-dessus les cellules (1, 2) et (2, 1) sont des vecteurs nuls. Les fichiers de fonctions exemples sont les suivants.

[EF/exemples/Assemblage_elementaire/exemple1_assemble_equ.m](#) (exemple 1)

```
%Fichier
```

```
% EF/exemples/Assemblage_elementaire/exemple1_assemble_equ :
```

```
% Utilisation de EF/Assemblage_elementaire/assemble_equ.
```

```
function L = exemple1_assemble_equ ;
```

```
T = [1,2,3] ;
```

```
X = [0,1,0] ;
```

```
Y = [0,0,1] ;
```

```
%Types d'espaces d'EF : F1 = P1
```

```
D.E = [1] ;
```

```

%Numero de la formule de quadrature sur les triangles :
D.intT = 4 ;
%Choix de l'espace produit : ici  $E = F_1 \times F_2 = E_1 \times E_2$ 
D.f = [1,1] ;
%Formules de quadrature :
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
                                assemble_coeff_T(X, Y, D) ;
%Valeur des fonctions de bases et de leurs derivees d'ordre 1 aux
%  points de la formule de quadrature. Coefficients de la formu-
%  le de quadrature :
for i=1 :length(D.f)
    I{i} = pre_assemble_T([], D.E(D.f(i)), [], [], [], [], [], ...
                        [], [], [], [], [], coeff, 1, [], [], [], []);
    for der=0 :2
        sc{i,der + 1} = pre_assemble_T([], D.E(D.f(i)), der, ...
            a1, b1, c1, a2, b2, c2, de, [], [], coeff, 3, D, ...
            L1, L2, L3);
        tc{i, der + 1} = sc{i,der + 1}' ;
    end ;
end ;
donnees = [] ;
uc = [] ;
i = [] ;
%Evaluation de la fonction forme bilineaire :
[Mcc, Ncc] = feval(@eq_Dom, donnees, uc, a1, b1, c1, a2, ...
                    b2, c2, de, Ac, Bc, i) ;
%Assemblage du systeme :
L = assemble_equ(D, I, Mcc, Ncc, sc, tc) ;
%Sous-fonction definissant la forme bilineaire :
function [Mc, Nc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, ...
                            c2, de, Ac, Bc, numK) ;
Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;
Un2 = {Un,Zero ; Zero,Un} ;

```

```

Zero2 = {Zero,Zero ;Zero,Zero} ;
Mc{1,1} = Un2 ;   Mc{1,2} = Zero2 ; Mc{1,3} = Zero2 ;
Mc{2,1} = Zero2 ; Mc{2,2} = Un2 ;   Mc{2,3} = Zero2 ;
Mc{3,1} = Zero2 ; Mc{3,2} = Zero2 ; Mc{3,3} = Un2 ;
Nc = Mc ;

```

EF/exemples/Assemblage_elementaire/exemple1b_assemble_equ.m (exemple 2)

```

%Fichier
%   EF/exemples/Assemblage_elementaire/exemple1b_assemble_equ :
%   Utilisation de EF/Assemblage_elementaire/assemble_equ.
%   Comme exemple1_assemble_equ, avec ici l'exemple d'un second
%   membre.
function L = exemple1b_assemble_equ ;
T = [1,2,3] ;
X = [0,1,0] ;
Y = [0,0,1] ;
%Types d'espaces d'EF : F1 = P1
D.E = [1] ;
%Numero de la formule de quadrature sur les triangles :
D.intT = 4 ;
%Choix de l'espace produit : ici E = F1 x F2 = E1 x E2
D.f = [1,1] ;
%Formules de quadrature :
[a1, b1, c1, a2, b2, c2, de, Ac, Bc, coeff, L1, L2, L3] = ...
                                assemble_coeff_T(X, Y, D) ;
%Valeur des fonctions de bases et de leurs derivees d'ordre 1 aux
%   points de la formule de quadrature. Coefficients de la formu-
%   le de quadrature :
for i=1 :length(D.f)
    I{i} = pre_assemble_T([], D.E(D.f(i)), [], [], [], [], ...,
                          [], [], [], [], [], coeff, 1, [], [], []);
    for der=0 :2
        sc{i,der + 1} = pre_assemble_T([], D.E(D.f(i)), der, ...,
                                         a1, b1, c1, a2, b2, c2, de, [], [], coeff, 3, D, ...

```

```

                                L1, L2, L3) ;

        tc{i, der +1}=sc{i,der + 1}';
    end ;
end ;
donnees = [] ;
uc = [] ;
i = [] ;
%Evaluation de la fonction forme lineaire :
[Mcc, Ncc, Fc] = feval(@eq_Dom, donnees, uc, a1, b1, c1, a2, ...
                                b2, c2, de, Ac, Bc, i) ;
%Traitement second membre fc. Mise dans un tableau de cellules de 3
%colonnes correspondant aux 3 scd membres. Les vecteurs lignes des
%valeurs aux pts de la form. de quad. sont transposes en colonnes.
for i=1 :2
    fc{i,1} = Fc{1}{i,1}';
    fc{i,2} = Fc{2}{i,1}';
    fc{i,3} = Fc{3}{i,1}';
end ;
%Assemblage du systeme :
L = assemble_equ(D, I, Mcc, Ncc, fc, tc) ;
L = [L{1,1} + L{1,2} ; L{2,1} + L{2,2}] ;
%Sous-fonction definissant la forme lineaire :
function [Mc, Nc, Fc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, ...
                                c2, de, Ac, Bc, numK) ;

Zero = zeros(1,length(Ac)) ;
Un = ones(1,length(Ac)) ;
Un2 = {Un,Zero ; Zero,Un} ;
Zero2 = {Zero,Zero ; Zero,Zero} ;
Mc{1,1} = Un2 ;   Mc{1,2} = Zero2 ; Mc{1,3} = Zero2 ;
Mc{2,1} = Zero2 ; Mc{2,2} = Un2 ;   Mc{2,3} = Zero2 ;
Mc{3,1} = Zero2 ; Mc{3,2} = Zero2 ; Mc{3,3} = Un2 ;
Nc = Mc ;
Fc{1} = {Ac.^2 ;Bc.^ 2} ;
Fc{2} = {Ac.^2 ;Bc.^ 2} ;

```

$$F_{c\{3\}} = \{A_c.^2; B_c.^2\};$$

Algorithme :

EF/Assemblage_elementaire/assemble_equ.m

On remet ici la partie déclarative de la fonction :

```
function L = assemble_equ(D, I, Mcc, Ncc, sc, tc);
```

Ci-dessous $E_i(K)$ est l'espace d'EF sur un triangle K correspondant au i -ème espace de l'espace EF produit E .

- Initialisation de J : $J\{i\}$ contient le vecteur ligne des coefficients d'intégration dupliqué sur $\dim(E_i(K))$ lignes :

```
for i=1 :length(D.f)
    J{i} = I{D.f(i)};
```

```
end ;
```

- Initialisation de L à une matrice de type CELL de taille $\text{size}(\text{tc}, 1) \times \text{size}(\text{sc}, 1)$. Le terme $L\{i, j\}$ est initialisé à la matrice nulle de taille $\text{size}(\text{tc}\{i, 1\}, 1) \times \text{size}(\text{sc}\{j, 1\}, 2)$.

Dans le cadre du code EF, L est une matrice de taille $n \times n$ dans le cas de l'assemblage de la forme bilinéaire liée au système et une matrice de taille $n \times n'$ dans le cas d'un second membre à valeur dans $\mathbb{R}^{n'}$, avant sommation suivant le deuxième indice dans les fonctions d'assemblage des seconds membres pour former le second membre par blocs (*cf.* Paragraphe 3.1.3.2).

Le terme $L\{i, j\}$ est alors une matrice de taille $\dim(E_i(K)) \times \dim(E_j(K))$ dans le premier cas et de dimension $\dim(E_i(K)) \times 1$ dans le second cas.

```
[s1,s2] = size(Mcc);
```

a) Cas de la forme factorisée :

- Initialisation de L :

```
if s1==1
    m1 = length(Mcc);
    m2 = length(Ncc);
    mf1 = size(tc,1);
    mf2 = size(sc,1);
    for i=mf1 :-1 :1
        stc(i, :) = size(tc{i});
    end;
    for i=mf2 :-1 :1
        ssc(i, :) = size(sc{i});
```

```

end ;
for i=mf1 :-1 :1
    for j=mf2 :-1 :1
        %0n a stc(i,1) = size(tc{j,1},1)
        %      ssc(j,2) = size(sc{j,1},2)
        L{i,j} = zeros(size(tc{j,1},1),size(sc{j,1},2)) ;
    end ;
end ;
• Calcul des matrices Mv et Nv avec la sous-fonction mult_cp_prep :
    for d1=m1 :-1 :1%derivation
        Mv{d1} = mult_cp_prep(Mcc{d1}) ;
    end ;
    for d2=m2 :-1 :1%derivation
        Nv{d2} = mult_cp_prep(Ncc{d2}) ;
    end ;

```

Ces matrices servent uniquement à accélérer les calculs en recherchant, dans la fonction `equationJ`, les termes *a priori* non nuls de la matrice $N'M'$.

```

• Calcul des contributions de la forme bilinéaire ou linéaire à L :
    for d1=1 :m1%derivation
        for d2=1 :m2%derivation
            L = equationJ(D, J, Mcc{d1}, Ncc{d2}, sc( :,d1), ...
                        tc( :,d2), L, Mv{d1}, Nv{d2}) ;
        end ;
    end ;

```

b) Cas de la forme non factorisée :

• Initialisation de L :

```

else
    mf1 = size(tc,1) ;
    mf2 = size(sc,1) ;
    for i = mf1 :-1 :1
        stc(i, :) = size(tc{i}) ;
    end ;
    for i = mf2 :-1 :1
        ssc(i, :) = size(sc{i}) ;
    end ;

```

```

end ;
for i=mf1 :-1 :1
    for j=mf2 :-1 :1
        %On a stc(i,1) = size(tc{j,1},1)
        %      ssc(j,2) = size(sc{j,1},2)
        L{i,j} = zeros(stc(i,1),ssc(j,2)) ;
    end ;
end ;
• Calcul des matrices Mv et Nv avec la sous-fonction mult_cp_prep :
    for d1=s1 :-1 :1%derivation
        for d2=s2 :-1 :1%derivation
            Mv{d1,d2} = mult_cp_prep(Mcc{d1,d2}) ;
        end ;
    end ;
    for d1=s1 :-1 :1%derivation
        for d2=s2 :-1 :1%derivation
            Nv{d1,d2} = mult_cp_prep(Ncc{d1,d2}) ;
        end ;
    end ;
• Initialisation de L :
    for d1=1 :s1%derivation
        for d2=1 :s2%derivation
            L = equationJ(D, J, Mcc{d1,d2}, Ncc{d1,d2}, ...
                sc( :,d1), tc( :, d2), L, Mv{d1,d2}, Nv{d1,d2}) ;
        end ;
    end ;
end%Fin du si
• Sous-fonction mult_cp_prep de la fonction assemble_equ :
function Mv = mult_cp_prep(M) ;
[m1,m2] = size(M) ;
for i=m1 :-1 :1
    for j=m2 :-1 :1
        Mv(i,j) = double(any( M{i,j} )) ;
    end ;
end ;
end ;

```

Fonction :

```
function uc = valeur_fonc_quad_K(donnees, Uc, Tc, a1, b1, c1, ...  
                                a2, b2, c2, numK) ;
```

La commande

```
uc = valeur_fonc_quad_K(donnees, Uc, Tc, a1, b1, c1, a2, b2, ...  
                        c2, numK) ;
```

donne la valeur de la solution et de ses dérivées aux points de la formule de quadrature. En développement, fonctionne. Non utilisé dans le code et les exemples.

Voir le commentaire dans le fichier :

EF_text/exemples/exemple_oldroyd_adapt1/equ/eq_normh_Dom_adapt.m

et son utilisation dans la version

EF_text/exemples/exemple_oldroyd_adapt1/equ/anc/eq_normh_Dom_adapt.m

qui utilise un ancien itéré de point fixe par la commande amenée par `Uc = donnees.Uanc`

Fonction de `trait_cb`.**Fonction :**

```
function
function [Nb, Sb] = cb(donnees, nomf, T, Arc, Pfc, Xc, Yc, ...
                      Dnc, Front) ;
```

La commande

```
[Nb, Sb] = cb(donnees, nomf, T, Arc, Pfc, Xc, Yc, Dnc, Front) ;
```

donne pour chaque espace de l'espace produit, à partir de la fonction désignée par `nomf` qui décrit les conditions au bord, les *ddl* sur lesquels on a une condition de type Dirichtlet et la valeur prise par les composantes de l'inconnue en ces *ddl*. En sortie, les *ddl* de l'inconnue *i* sont donnés par le vecteur colonne `Nb{i}` et les valeurs correspondantes par le vecteur colonne `Sb{i}`. Ci-dessus, `donnees` est la variable qui décrit la structure de données du problème EDP, `T` et `Front` sont des variables de la structure de données du maillage, `T` décrit les sommets du maillage triangle par triangle et `Front`, variable de type `CELL ARRAY`, décrit la frontière du domaine et sa partition en morceaux. Ensuite `Arc`, `Pfc`, `Xc`, `Yc` sont des variables de type `CELL ARRAY` qui appartiennent à la structure de données relative aux EF et qui sont calculées en général par les fonctions `EF_prepa` ou `EF_prepa_don`. La variable `Arc` est telle que `Arc{i}` décrit arête par arête les *ddl* situés sur les arêtes du *i*-ème espace d'EF de l'espace EF produit *E*, `Xc{i}` et `Yc{i}` les coordonnées de ces *ddl*, `Pfc` décrit les arêtes du maillage situées sur la frontière à partir de leur numérotation locale sur chaque morceau de frontière. Enfin `Dnc` est le vecteur ligne de type `DOUBLE` tel que `Dnc(i)` est la dimension de l'espace *i* de l'espace produit. `Dnc` est donné par `Dnc = Ed(donnees.D.f)` où `Ed` est stocké dans le fichier de données relatif aux espaces d'EF.

Variables d'entrée :

`donnees` : `STRUCT ARRAY`. Variable décrivant les données du problème EF.

`nomf` : `FUNCTION_HANDLE ARRAY` sur la fonction définissant les conditions au bord

`T` : `DOUBLE ARRAY`. Description triangle par triangle des numéros de sommets du maillage, cf. Tableau 2.2.2-1, Paragraphe 2.2.2.

`Arc` : `CELL ARRAY`. Vecteur ligne. `Arc{i}` est une matrice de type `DOUBLE ARRAY` décrivant les *ddl* arêtes par arêtes du *i*-ème type d'espace d'EF choisi, cf. le Tableau 2.2.3-1b du Paragraphe 2.2.3, `Arc{i}` est un vecteur colonne nul dans le cas d'un espace d'EF qui n'a pas de *ddl* d'arêtes comme les espaces d'EF discontinus de numéros 100 à 199 et de numéro 300. La valeur de `Arc{i}` est donnée par la valeur de la variable `Ad` calculée dans `maillage_ddl` dans le cas du *i*-ème type d'espace d'EF choisi.

`Pfc` : `CELL ARRAY`. Vecteur colonne de cellules de longueur égale au nombre de morceaux de frontière du domaine. Cette variable a donc le même nombre de

cellules que la variable de maillage **Front** (*cf.* Tableau 2.2.2-1, Paragraphe 2.2.2). Sur chaque morceau de frontière, les arêtes reçoivent une numérotation locale allant de 1 au nombre d'arêtes sur ce morceau de frontière. La cellule **Pfc**{*i*} est un tableau de trois colonnes de type **DOUBLE ARRAY** qui décrit les arêtes du morceau de frontière numéro *i* de la manière suivante. Soit une arête de numéro *j* dans sa numérotation sur le morceau de frontière numéro *i*, alors **Pfc**{*i*}(*j*, 1) est le numéro de triangle auquel appartient cette arête, **Pfc**{*i*}(*j*, 2) son numéro d'arête localement sur ce triangle (valeur parmi 1, 2 ou 3) et **Pfc**{*i*}(*j*, 3) son numéro global d'arête dans la numérotation globale des arêtes. **Pf** vaut [] si **Front** = []. Le calcul de cette variable est réalisé par `maillage_triangle_arete_ps` à partir de **Front** dans l'ordre des arêtes de **Front**{1}, **Front**{2} *etc.* Pour cette variable se reporter aussi au Tableau 2.2.3-1b du Paragraphe 2.2.3.

Xc, **Yc** : **CELL ARRAY**. Vecteurs lignes de cellules. Les cellules sont des vecteurs lignes de type **DOUBLE ARRAY**. Les vecteurs **Xc**{*i*} et **Yc**{*i*} sont les vecteurs des coordonnées en *x* et *y* des *ddl* du *i*-ème type d'espace d'EF choisi, *cf.* Tableau 2.2.3-1b du Paragraphe 2.2.3.

Dnc : **DOUBLE ARRAY**. Vecteur ligne. **Dnc**(*i*) est la dimension du *i*-ème espace d'EF de l'espace EF produit *E*. Dans cette fonction **Dnc** peut aussi être un vecteur colonne.

Front : **CELL ARRAY**. Vecteur colonne. La cellule *i* de **Front** est un vecteur ligne de type **DOUBLE ARRAY** qui décrit consécutivement sur la frontière les numéros de sommets du morceau de frontière numéro *i*, *cf.* Tableau 2.2.2-1, Paragraphe 2.2.2.

Variables de sortie :

Nb : **CELL ARRAY**. **Nb**{*i*} : est soit vide soit un vecteur colonne de **DOUBLE ARRAY**. Dans ce cas **Nb**{*i*} est un vecteur qui contient les numéros de *ddl* sur lesquels on a une condition de Dirichtlet pour la variable *i* et ce vecteur ne contient pas d'autres termes. Le vecteur **Nb**{*i*} peut contenir plusieurs fois le même *ddl*, par exemple lorsque celui est commun à deux frontières

Sb : **CELL ARRAY**. **Sb**{*i*} : Vecteur colonne **SPARSE ARRAY** de taille $\dim(E_i) = \mathbf{Dnc}(i)$, *i*-ème espace d'EF de l'espace EF produit *E*. Valeurs de la composante de la variable *i* au bord (Dirichtlet). La valeur de la composante au *ddl* numéro **Nb**{*i*}(*j*) est donnée par **Sb**{*i*}(**Nb**{*i*}(*j*)) et **Sb**{*i*}(*k*) est nul pour les autres *ddl* *k* qui n'appartiennent pas au vecteur **Nb**{*i*}.

Exemples : Un exemple pour la prise en compte des conditions de type normale est donnée au Paragraphe 2.10.7. Un exemple est donné dans le fichier `exemple_cb` du répertoire `EF/exemples/trait_cb`. Dans cet exemple l'espace EF produit *E* est donné par

$$E = E_1 \times E_2 \times E_3 \times E_4 \times E_5 = P_1^c \times P_3^c \times P_3^c \times (P_2 + 3 \text{ bulles } P_4) \times P_2^d,$$

avec le maillage par 8 triangles du carré unité *C* de la Figure 2.3.2-1. La frontière du carré est partitionnée en 4 morceaux de frontière numérotés de 1 à 4 qui sont

$\{(x, y) \in C, y = 1\}$, $\{(x, y) \in C, x = 1\}$, $\{(x, y) \in C, y = 0\}$ et $\{(x, y) \in C, x = 1\}$. Les sommets de la frontière sont parcourues dans le sens trigonométrique inverse dans la variable **Front** (*cf.* cette variable au Tableau 2.2.2-1, Paragraphe 2.2.2).

Les conditions au bord sont les suivantes

$E_1 : u_1(x, y) = x + y$ sur les frontières 1, 2, 3 et 4 du domaine,

$E_2 : u_2(x, y) = x - y$ sur la frontière 1 du domaine,

E_3 : pas de conditions au bord sur u_3 ,

$E_4 : u_4(x, y) = x + y$ sur les frontières 1 et 3 du domaine,

$E_5 : u_5(x, y) = x + y$ sur les frontières 1 et 2 du domaine. Comme il s'agit d'un espace d'EF discontinu, l'ensemble des *ddl* est vide pour ce type d'espace.

Cet exemple est mis en œuvre par la commande

`[Nb, Sb, Xc, Yc] = exemple_cb`

Dans cet exemple les types d'espaces sont $F_1 = P_1^c$, $F_2 \times P_3^c$, $F_3 = (P_2 + 3 \text{ bulles } P_4)$, $F_4 = P_2^d$ et on pose $E = F_1 \times F_2 \times F_3 \times F_4$, l'espace EF produit.

Les valeurs de **Nb** les suivantes :

$$\mathbf{Nb}\{1\} = [7, 8, 9, 3, 6, 9, 1, 2, 3, 1, 4, 7]',$$

$$\mathbf{Nb}\{2\} = [7, 8, 9, 38, 39, 40, 41]',$$

$$\mathbf{Nb}\{3\} = [],$$

$$\mathbf{Nb}\{4\} = [7, 8, 9, 24, 25, 1, 2, 3, 10, 13]',$$

$$\mathbf{Nb}\{5\} = [].$$

Ces *ddl* ont leurs coordonnées cartésiennes données par $\mathbf{Xc}\{\mathbf{D.f}(i)\}(\mathbf{Nb}\{i\})$ et $\mathbf{Yc}\{\mathbf{D.f}(i)\}(\mathbf{Nb}\{i\})$ $i = 1, \dots, 5$, avec $\mathbf{D.f} = [1, 2, 2, 3, 4]$ (*cf.* Tableau 2.2.1-1c du Paragraphe) définissant l'espace EF produit E à partir de la liste des types d'espaces d'EF utilisés donnée par $\mathbf{D.E} = [1, 3, 24, 102]$ (*cf.* aussi le Tableau 2.2.1-1c) :

$$\mathbf{Xc}\{1\}(\mathbf{Nb}\{1\}) = [0, 1/2, 1, 1, 1, 1, 0, 1/2, 1, 0, 0, 0],$$

$$\mathbf{Xc}\{2\}(\mathbf{Nb}\{2\}) = [0, 1/2, 1, 1/3, 1/6, 5/6, 2/3],$$

$$\mathbf{Xc}\{2\}(\mathbf{Nb}\{3\}) = [],$$

$$\mathbf{Xc}\{3\}(\mathbf{Nb}\{4\}) = [0, 1/2, 1, 1/4, 3/4, 0, 1/2, 1, 1/4, 3/4],$$

$$\mathbf{Xc}\{4\}(\mathbf{Nb}\{5\}) = []$$

et

$$\mathbf{Yc}\{1\}(\mathbf{Nb}\{1\}) = [1, 1, 1, 0, 1/2, 1, 0, 0, 0, 0, 1/2, 1],$$

$$\mathbf{Yc}\{2\}(\mathbf{Nb}\{2\}) = [1, 1, 1, 1, 1, 1, 1],$$

$$\mathbf{Yc}\{2\}(\mathbf{Nb}\{3\}) = [],$$

$$\mathbf{Yc}\{3\}(\mathbf{Nb}\{4\}) = [1, 1, 1, 1, 1, 0, 0, 0, 0, 0].$$

$$\mathbf{Yc}\{4\}(\mathbf{Nb}\{5\}) = [].$$

Les valeurs trouvées de **Sb** sont

$$\begin{aligned} \text{Sb}\{1\}(\text{Nb}\{1\}) &= [1, 3/2, 2, 1, 3/2, 2, 0, 1/2, 1, 0, 1/2, 1]', \\ \text{Sb}\{2\}(\text{Nb}\{2\}) &= -[1, 1/2, 0, 2/3, 5/6, 1/6, 1/3]', \\ \text{Sb}\{3\}(\text{Nb}\{3\}) &= [], \\ \text{Sb}\{4\}(\text{Nb}\{4\}) &= [1, 3/2, 2, 5/4, 7/4, 0, 1/2, 1, 1/4, 3/4]', \\ \text{Sb}\{5\}(\text{Nb}\{5\}) &= []. \end{aligned}$$

Les *ddl* des espaces finis, ainsi que ceux sur lesquels portent les conditions au bord (*ddl* désignés par **Nb**) sont représentés à la figure 4.3-17 ci-dessous.

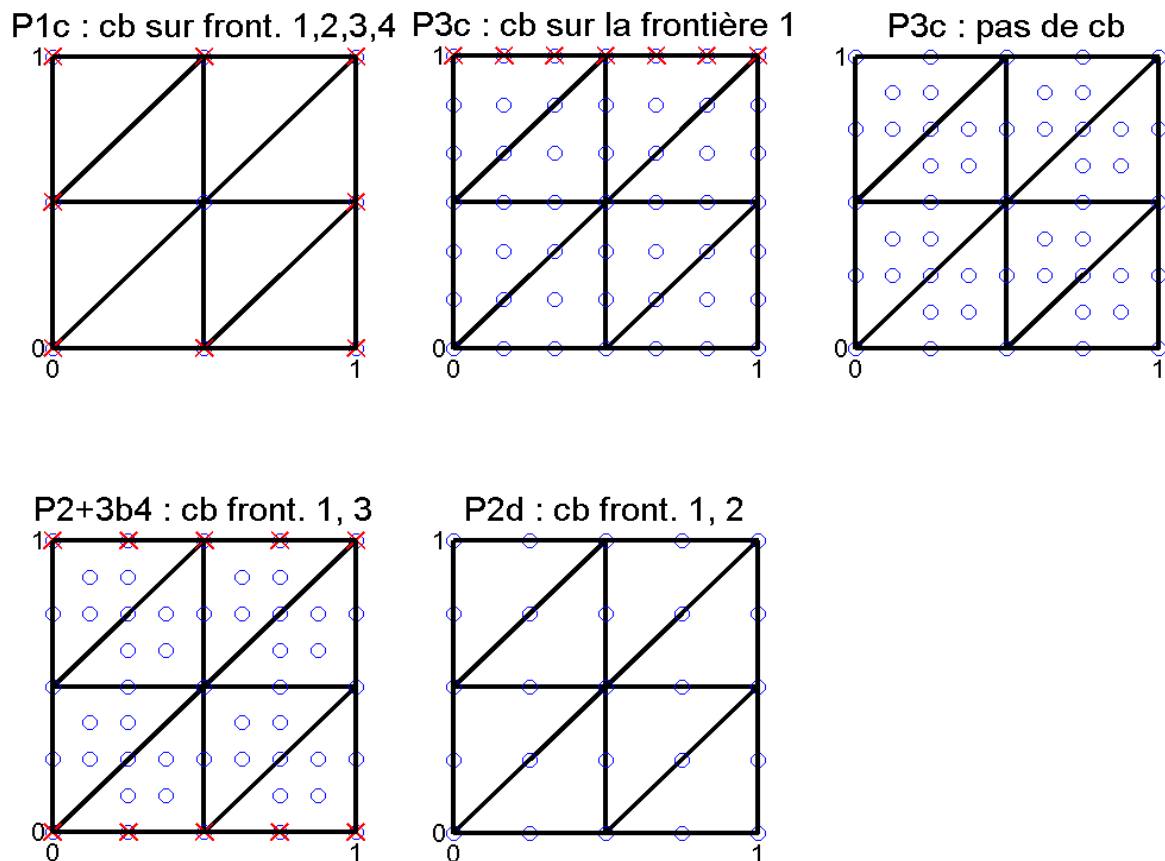


Figure 4.3-17. Les *ddl* sur lesquels portent les conditions au bord sont marquées d'une croix. Le P_2^d n'a pas de *ddl* sur la frontière.

Dans cet exemple la fonction qui définit les conditions au bord et désignée par **nomf** dans les variables d'entrée de la fonction **cb** est donnée par :

```
function [fb,sb] = cb_don(donnees)
```

```

%Frontieres avec conditions au bord (cb) imposees
fb{1} = [1 :4] ; %cb sur les frontieres 1,2,3,4 pour l'inconnue
%
%                                numero 1
fb{2} = 1 ;    %cb sur la frontiere 1, pour l'inconnue numero 2
fb{3} = [] ;   %pas de cb pour l'inconnue numero 3
fb{4} = [1,3] ; %cb sur les frontieres 1 et 3 pour l'inconnue
%
%                                numero 2
fb{5} = [1 :2] ; %cb sur les frontieres 1,2,3,4 pour l'inconnue
%
%                                numero 1 (EF discontinu)
%Fonction donnant la valeur sur les frontieres :

%On met la fonction nulle pour tous les espaces et frontieres
% (facultatif)
for i=1 :5
sb.v{1,i} = @f1 ;
sb.v{2,i} = @f1 ;
end ;

%Inconnue 1, frontiere 1
sb.v{1,1} = @f2 ;
%Inconnue 1, frontiere 2
sb.v{1,2} = @f2 ;
%Inconnue 1, frontiere 3
sb.v{1,3} = @f2 ;
%Inconnue 1 frontiere 4
sb.v{1,4} = @f2 ;
%Inconnue 2, frontiere 1
sb.v{2,1} = @f3 ;
%Inconnue 4, frontiere 1
sb.v{4,1} = @f2 ;
%Inconnue 4, frontiere 3
sb.v{4,3} = @f2 ;
%Inconnue 5 frontiere 1
sb.v{5,1} = @f2 ;

```

```
%Inconnue 5 frontiere 2
sb.v{5,2} = @f2;
function z = f1(x,y)
z = 0.*x;
%entree
function z = f2(x,y);
z = x + y;
function z = f3(x,y);
z = x - y;
```

Algorithme :

EF/algo/algo/cb.m

On remet ici la partie déclarative de la fonction :

```
function [Nb, Sb] = cb(donnees, nomf, T, Arc, Pfc, Xc, Yc, ...
                                Dnc, Front);
```

- Evaluation des variables de sortie de la fonction définissant les conditions au bord.

```
[fb,sb] = feval(nomf,donnees);
```

Ci-dessus `fb` est de type `CELL ARRAY`, `fb{i}` est un vecteur ligne qui indique les numéros des frontières sur lesquelles on a une condition aux limites pour la i -ème variable. Cette variable est décrite dans le tableau 2.2.4-2 a) du Paragraphe 2.2.4.

Ensuite `sb` est une variable de type `STRUCT ARRAY` décrite dans le tableau 2.2.4-2 a) b) et c) du Paragraphe 2.2.4. Les champs de cette variable utilisés par le code EF sont les champs `sb.v`, `sb.sbc` qui sont aussi utilisés par `cb` et les champ `sb.un` et `sb.a` utilisés en particulier par la fonction `EF_assemblage`.

Le terme `sb.v{i,j}` est un **handle** sur la fonction de deux variables qui définit la condition aux limites pour la i -ème variable sur la frontière numéro `fb{i}(j)`.

Lorsque les fonctions définissant les conditions au bord contiennent des paramètres on pourra utiliser des variables globales ou bien utiliser un terme `sb.v{i,j}` qui soit un tableau de deux cellules dont la première contient le handle de fonction du cas précédent et la seconde cellule le nombre de variables d'entrée de la fonction définissant les conditions au bord. Ce nombre vaut soit 2, les variables d'entrée étant x et y les coordonnées cartésiennes des points frontières ou bien 3 avec la variable supplémentaire `donnees` pour le passage du paramètre. Un exemple d'utilisation de paramètres est donné dans la résolution du Problème ($\mathcal{P}_{6h}^{(7)}$) du Paragraphe 2.10.10.1 avec la commande

```
cd C :/EF_text/exemples/exemple_stokes_puis, instruct1
```

pour le passage du paramètre r de la loi puissance par variable globale et par

```
cd C :/EF_text/exemples/exemple_stokes_puis, instruct1_cell
```

pour le passage de la valeur de r par l'intermédiaire de la variable `donnees`, et son champ du champ utilisateur `donnees.r`, passée à la sous-fonction `f2`, qui a alors 3 variables d'entrée, de la fonction `cb_don_cell.m` définissant les conditions au bord.

Le champ `sb.un` est relatif aux conditions normales et tangentes et le champ `sb.a` est relatif aux conditions du type $AX = b$. Ces deux champs ne sont pas utilisés dans `cb`, ils interviennent dans la fonction `EF_assemblage` où ils sont appelés respectivement par les fonctions `changt_base_nt` `changt_base_a`.

Le champ `sb.cbc` est facultatif. Il est utilisé par la fonction `cb` et permet de choisir une condition au bord sur les *ddl* à l'intersection de deux frontières lorsque deux conditions différentes sont données en ce point. Un exemple d'utilisation du champ `sb.cbc` est donné dans le traitement du problème $(\mathcal{P}_6^{(4')})$ du Paragraphe 2.10.7 avec la commande

```
cd C :/EF_text/exemples/exemple_stokes_rotation, instruct2,
```

et le fichier `cb_dontheta2.m` définissant les conditions au bord.

Ci-dessous, le cas [1] : `sb.v{i,j}` est un tableau de type de type `CELL ARRAY` et le cas [2] : `cbc` est un champ de `cb` sont traités dans la fonction `cb` et sont en cours de développement. Il ne sont pas détaillés dans l'algorithme qui suit.

- Calcul des variables de sortie : Si `fb` est vide on pose

```
Nb = [] ;
```

```
Sb = [] ;
```

et l'algorithme se termine. On n'a alors pas de conditions aux limites.

Sinon on pose

```
s = length(Front) ;
```

le nombre de morceaux de frontières

- Variable décrivant les espaces d'EF utilisés :

```
D = donnees.D ;
```

- Boucle sur chaque type d'espace pour déterminer les *ddl* sur la frontière par type d'espaces et par numéro de frontière. Ci-dessous `F2 = Arc{i}` est la description arêtes par arêtes du maillage des *ddl* du i -ème type d'espace d'EF F_i choisi, ensuite `Pfc{j}(:,3),:` désigne les numéros d'arêtes globaux des arêtes de la frontière numéro j .

La variable `Z` calculée ci-dessous est alors une variable de type `CELL ARRAY` telle que `Z{i,j}` ci-dessous représente les *ddl* de F_i placés sur la frontière numéro j .

```
%ddl pour chaque type d'espace
```

```
for i=1 :length(D.E)
```

```

F2 = Arc{i};
    for j=1 :s(1)
        %ddl mis a 0 si pas de ddl
        Z{i,j} = F2(Pfc{j}( :,3), :);
        Z{i,j} = union(Z{i,j}( :),Z{i,j}( :));
        if Z{i,j}==0;
            Z{i,j} = [] ;%pas de ddl, ci-apres
            %             les affectations ne font rien
            %             pour cette valeur []
        end;
    end;
end;
• Initialisation de Sb et Nb
for i=1 :length(D.f)
    Sb{i,1} = sparse(Dnc(i),1) ;%vecteur colonne
    Nb{i,1} = [] ;
end;
• Calcul de Sb et Nb. Nb{i} ci-dessous contient alors les ddl de  $E_i$  le i-ème espace
de l'espace EF produit  $E$  et Sb{i} les valeurs imposées sur les composantes sur
 $E_i$  de la solution en ces ddl, avec la valeur 0 pour les autres ddl.

if ~isfield(sb,'cbc')
    for i=1 :length(fb)%ieme inconnue espace Ei de numero
        %                                     D.E(D.f(i))
        for j=fb{i}%Frontiere j
            Nb{i} = [Nb{i}; Z{D.f(i),j}];
            if iscell(sb.v{i,j});
                [1] en développement, non détaillé. fonctionne.
            else
                Sb{i}(Z{D.f(i),j}) = feval( sb.v{i,j},...
                    Xc{D.f(i)}( Z{D.f(i),j} ), Yc{D.f(i)}( Z{D.f(i),j} ) );
            end;
        end;
    end;
else
    [2] en développement, non détaillé. fonctionne.
end;

```


5.1. APPENDICE. AJOUT DES REPERTOIRES DU CODE AU PATH.

5.1. Ajoût des répertoires du code aux chemins de recherche.

Au début de l'utilisation du code. Ajouter les répertoires du code aux chemins de recherche.

Si le chemin du répertoire EF est C :/EF : les commandes suivantes

```
cd C :/EF
```

```
path_EF
```

ajoutent les répertoires du code au path. La commande path donne alors

```
>> path
```

```
MATLABPATH
```

```
C : \EF\element_t
```

```
C : \EF\adapt
```

```
C : \EF\trait_maillage
```

```
C : \EF\exemples\trait_cb
```

```
C : \EF\exemples\trace_graphe_interp
```

```
C : \EF\exemples\Assemblage_elementaire
```

```
C : \EF\exemples\Assemblage
```

```
C : \EF\exemples
```

```
C : \EF\trace_graphe_interp
```

```
C : \EF\utilitaires
```

```
C : \EF\trait_cb
```

```
C : \EF\mat_cell
```

```
C : \EF\deg_lib
```

```
C : \EF\Assemblage_elementaire
```

```
C : \EF\Assemblage
```

```
C : \EF
```

etc. (plus les autres repertoires du path hormis ceux du code EF)

Si le répertoire EF a une autre chemin, on modifiera en conséquence la valeur de la variable `racine` dans le fichier *EF/path_EF.m* qui débute par :

Fichier C :/EF/path_EF.m

```
%A modifier suivant l'environnement
```

```
%Dossiers utilises
```

```
function path_EF
```

```
racine='C :/EF' ;
```

5. APPENDICE

```
function path_EF
racine='C :/EF' ;
```

Utilisation des fichiers exemples du code.

Pour l'utilisation des fichiers exemples du répertoire :

```
C :/EF_text/exemples
```

on modifiera aussi en conséquence les chemins de répertoires utilisés dans les fichiers exemples.

Par exemple, si on reprend l'exemple du Paragraphe 2.1.4 donné par la commande de résolution :

```
cd C :/EF_text/exemples/exemple1_Laplacien, instruct1
```

et que nous plaçons le répertoire `exemple1_Laplacien`, dans le répertoire `D :/` sous le nom, à titre d'exemple, `exemple1_Laplacien.D`. La commande d'appel du fichier *instruct1.m* de ce répertoire est alors

```
cd D :/exemple1_Laplacien_D, instruct1
```

• **Modification du fichier d'instructions.** Pour la prise en compte du changement de répertoire, le début du fichier *instruct1.m* donné par :

Fichier C :/EF_text/exemples/exemple1_Laplacien/instruct1.m

```
clear
close all
rep = pwd ;
%Repertoires a ajouter au path :
racine = 'C :' ;
cd([racine,'/EF']);
path_EF ;
%Nouveau repertoire courant : repertoire de la fonction d'initiali-
% sation de la variable de type struct qui decrit les donnees :
cd([racine,'/EF_text/exemples/exemple1_Laplacien/donnees']);
%Initialisation de la variable "donnees" :
fdon = @don1 ;
donnees = feval(fdon) ;
cd(rep) ;
```

est alors modifié par

5.1. APPENDICE. AJOUT DES REPERTOIRES DU CODE AU PATH.

Fichier D :/exemple1-Laplacien/instruct1.m

```
clear
close all
rep = pwd ;
%Repertoires a ajouter au path :
racine = 'C :' ;
cd([racine,'/EF']);
path_EF ;
%Nouveau repertoire courant : repertoire de la fonction d'initiali-
% sation de la variable de type struct qui decrit les donnees :
cd(['D :/exemple1-Laplacien_D/donnees']);
%Initialisation de la variable "donnees" :
fdon = @don1 ;
donnees = feval(fdon) ;
cd(rep) ;
```

ainsi que la commande

Fichier C :/EF_text/exemples/exemple1-Laplacien/instruct1.m

```
save(['C :/EF_text/exemples/exemple1-Laplacien/resultats', ...
      '/sol_don1'], 'donnees', 'Uc', 'T', 'X', 'Y', 'Front') ;
```

est remplacée par la commande

Fichier D :/exemple1-Laplacien_D/instruct1.m

```
save(['D :/exemple1-Laplacien_D/resultats', ...
      '/sol_don1'], 'donnees', 'Uc', 'T', 'X', 'Y', 'Front') ;
```

• **Modification du fichier de description des données.** Ensuite le fichier de donnée appelé par *instruct1.m* et qui commence par :

Fichier C :/EF_text/exemples/exemple1-Laplacien/donnees/don1.m

```
function donnees = don1 ;
%DON1 Initialisation de donnees definissant un probleme EF.
% Ci-dessous les champs utilises resp. par EF_Prepa et
% EF_assemblage sont indiqués par resp. (1) et (2).

%Chemin du repertoire contenant les donnees :
nom_rep = 'C :/EF_text/exemples/exemple1-Laplacien' ;
```

5. APPENDICE

est modifié par

Fichier D :/exemple1_Laplacien_D/donnees/don1.m

```
function donnees = don1 ;
%DON1 Initialisation de donnees definissant un probleme EF.
%   Ci-dessous les champs utilises resp. par EF_Prep et
%   EF_assemblage sont indiqués par resp. (1) et (2).

%Chemin du repertoire contenant les donnees :
nom_rep = 'D :/exemple1_Laplacien_D' ;
```

On trouvera le répertoire exemple `exemple1_Laplacien_D` dans le répertoire

```
C :/EF_text/exemples ;
```

Il ne contient que les fichiers nécessaires au calcul par la commande

```
cd D :/exemple1_Laplacien_D, instruct1
```

En général, comme dans l'exemple ci-dessus, les fichiers à modifier seront le fichier d'instructions et le fichier décrivant les données.

5.2. Variables du code, récapitulatif de tableaux.

Nous commençons par la variable de type `struct`, que nous appellerons `donnees`, qui décrit les données du problème EDP et de son approximation. Les champs de cette variable réservés par le code sont décrits aux Tableaux 2.2.1-1a, b, c et d. Le Tableau 2.2.1-1a décrit les champs contenant des noms de fichiers `.mat` qui sont le fichier contenant le maillage et le fichier de données intrinsèques aux espaces d'EF.

Champs réservés de la variables décrivant les données	
Noms de fichiers <code>.mat</code>	
<code>nom_mail</code>	Fichier <code>.mat</code> contenant le maillage.
<code>nom_prep</code>	Fichier <code>.mat</code> de données relatives aux espaces d'EF et aux formules d'intégration.

Tableau 2.2.1-1a.

Les champs du Tableau 2.2.1-1b contiennent des noms de fichiers `.m` qui décrivent les formes linéaires et bilinéaires de (1.5) et (1.10), excepté la méthode de Lesaint-Raviart (*cf.* Tableau 2.2.1-1d), ainsi que les conditions aux limites.

Champs réservés de la variables décrivant les données. Suite.	
Handles sur des fonctions (champs facultatifs). Les champs de ce tableau sont des handles sur des fonctions. Ils sont facultatifs ou peuvent valoir <code>[]</code> .	
<code>feq_Dom</code>	Fonction décrivant les EDP sur le domaine. Correspond à a_Ω .
<code>feq_Fr</code>	Fonction décrivant les EDP sur les frontières. Correspond à a_Γ .
<code>feq_Ad</code>	Fonction décrivant les EDP sur les arêtes du maillage. Correspond à a_Σ . En développement. Fonctionne.
<code>fscm_Dom</code>	Fonction décrivant le second membre (scm) sur le domaine. Correspond à b_Ω .
<code>fscm_Fr</code>	Fonction décrivant le second membre sur les frontières. Correspond à b_Γ .
<code>fscm_Ad</code>	Fonction décrivant le second membre sur les arêtes du maillage. Correspond à b_Σ . En développement, fonctionne.
<code>fc_b</code>	Fonction décrivant les conditions aux limites.

Tableau 2.2.1-1b.

Ci-après nous donnons la description du champ D de la variable `donnees`.

5. APPENDICE

Champs réservés de la variables décrivant les données. Suite.	
Champ D. Description des EF. Ce champ est lui même une variable de type struct . Dans la colonne de droite D est mis pour donnees.D .	
D.E	Double array. Vecteur ligne. Choix des types d'espaces d'EF. Exemple : le choix P_1^c , P_3^c et P_2^c (espaces numéros 1, 3 et 2, cf. la fin du Paragraphe 1.4) est indiqué par $D.E = [1, 3, 2]$.
D.intT	Double array. Scalaire. Choix de la formule d'intégration sur les triangles. Choix actuellement programmés : 4, 5, 6, 8, 10 (ces nombres correspondent à l'ordre des méthodes).
D.intA	Double array. Scalaire. Choix de la formule d'intégration sur les arêtes. Choix actuellement programmés : 4, 6, 10 (l'ordre de ces méthodes est respectivement 5, 7, 11).
D.der	Scalaire. Vecteur ligne. Dérivations utilisées : 0, x , y . Dans la version actuelle du code ce champ est toujours fixé à la valeur correspondante à 0, x , y qui est $[0, 1, 2]$.
D.f	Double array. Vecteur ligne. Choix de l'espace produit. $D.f(i)$ indique que le i -ème espace de l'espace produit est l'espace n° $D.E(D.f(i))$. Ainsi, avec l'exemple, si $D.f = [1, 2, 3, 2]$ on a $D.E(D.f) = [1, 3, 2, 3]$ ce qui donne $E = P_1^c \times P_3^c \times P_2^c \times P_3^c$.
D.inc_cell	Double array. Matrice de deux lignes ou $[]$. Elimination d'inconnues (par cellules). L'inconnue n° $D.inc\_cell(2, i)$ est éliminée en supposant qu'elle vérifie les mêmes équations que l'inconnue $D.inc\_cell(1, i)$. Les inconnues désignées par chaque colonne de D.inc_cell doivent être dans le même espace d'EF. Les lignes 1 et 2 doivent former des ensembles disjoints. Exemple : $D.inc\_cell = [5, 5; 1, 2]$: signifie qu'on élimine les inconnues n° 1 et 2 et qu'elles vérifient la même équation que l'inconnue n° 5. On ne pourra pas mettre $D.inc\_cell = [5, 1; 1, 2]$ car les lignes sont non disjointes. Champ facultatif.
D.inc_point	Cell array. Tableau $p \times 3$ avec $1 \leq p \leq n$ ou $[]$. Fonctions de base EF à enlever. $D.inc\_point\{i, 1\}$: numéro d'inconnue. Cellule $\{i, 2\}$: vecteur ligne (double) contenant les numéros des fonctions de base à éliminer. Cellule $\{i, 3\}$: vecteur ligne (double) de même taille que le précédent, valeur fixant les coefficients de ces fonctions de base. $D.inc\_point\{i, 2\}$ peut aussi prendre la valeur 'end' qui indique que la dernière fonction de base est éliminée, le vecteur $D.inc\_point\{i, 3\}$ est alors un scalaire. Champ facultatif.

Tableau 2.2.1-1c.

Champs réservés de la variables décrivant les données. Suite et Fin.	
Champ D. Suite et Fin. Méthode de Lesaint-Raviart	
D.LR	Champ facultatif. Tableau de cellules $p \times 2$ avec $1 \leq p$ ou bien vaut $[]$. La première colonne contient des handles de fonctions. La deuxième des vecteurs lignes. Dérivation suivant la méthode de Lesaint-Raviart. Exemple : $D.LR = \{ @fu1, [1, 3]; @fu2, [2, 4] \}$ indique que les variables 1 et 3 seront dérivées selon (1.17) avec le champ de vecteurs u_1 donné par $@fu1$ et les variables 2 et 4 par le champ de vitesse u_2 donné par $@fu2$. Dans (1.17) on a alors $M_1 = \{1, 3\}$ et $M_2 = \{2, 4\}$.
Syntaxe des fonctions de D.LR	Les fonctions sur lesquelles portent les handles le première colonne de D.LR ont la syntaxe suivante : $[Mc, Nc] = fu1(donnees, numF, uc, a1, b1, c1, \dots$ $\quad a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;$ où les variables sont décrites au Tableau 2.2.4-1 du Paragraphe 2.2.4. La variable de sortie Mc (resp. Nc) est un vecteur ligne qui est la valeur aux points de coordonnées cartésiennes données par (Ac, Bc) de la première (resp. deuxième) composante du champ de vecteur (ce vecteur ligne a donc même taille que Ac et Bc).

Tableau 2.2.1-1d.

Le fichier de maillage d'extension *.mat* dont le nom est indiqué par le champ **nom_mail** contient les variables de maillage décrites dans le Tableau 2.2.2-1 qui suit.

Les Tableaux 2.2.3-1a, 2.2.3-1b suivants décrivent le contenu du fichier désigné par la variable **donnees.nom_prep** : le Tableau 2.2.3-1a décrit les variables qui contiennent des termes relatifs aux matrices d'assemblage d'élémentaire et le Tableau 2.2.3-1b, décrit les *ddl* des espaces d'éléments finis, voici un aperçu des variables de ce tableau :

Les variables **Pa**, **Pf**, et **Pfc** sont de nature géométrique et sont relatives à une numérotation des arêtes du maillage effectuée dans **EF_Prepa** :

Pa : description des arêtes du maillage.

Pf : description des arêtes du maillage situées sur la frontière.

Pfc : comme **Pf** avec une description morceau de frontière par morceau frontière.

Ensuite les variables **Tc**, **Arc**, **Xc**, **Yc**, sont des tableaux de cellules dont chaque cellule est relative aux *ddl* de l'espace d'EF F_i , l'espace d'EF dont le numéro est donné par **donnees.D.E(i)** :

5. APPENDICE

$Tc\{i\}$: Description triangle par triangle des *ddl* situés sur les triangles de l'espace F_i .

$Xc\{i\}$: coordonnées en x des *ddl* de l'espace d'EF F_i .

$Yc\{i\}$: coordonnées en y des *ddl* de l'espace d'EF F_i .

$Arc\{i\}$: description arête par arête des *ddl* situés sur les arêtes de l'espace d'EF F_i .

Les valeurs prises par les variables ci-dessus sont décrites au Paragraphe 2.3 dans le cas d'un simple triangle et dans le cas d'un maillage d'un carré par 8 triangles.

Il est à noter que la connaissance des variables stockées dans ce fichier n'est pas nécessaire pour utiliser le code, elle intervient dans le cas où l'on veuille modifier le code comme par exemple ajouter de nouveaux éléments finis.

Pour les tracés, l'interpolation des solutions, on pourra avoir à utiliser, outre les variables de maillage T , X , Y les variables de *ddl* Tc , Xc , Yc .

Les variables de maillage sont les suivantes :

Structures de données du fichier de maillage	
Front	Cell array. Vecteur colonne. Description des frontières. Front est un vecteur colonne de cellules. La frontière est partitionnée en morceaux de frontière. $Front\{i\}$ est un vecteur ligne qui contient les numéros de sommets du maillage placés sur le morceau de frontière n° i . Les termes consécutifs de ce vecteur doivent être des numéros de sommets consécutifs sur la frontière. Si la frontière i se referme, le premier et le dernier terme de $Front\{i\}$ doivent être identiques. On peut aussi poser $Front = []$.
Noeu	Double array. Scalaire. Nombre de sommets du maillages.
T	Double array. Matrice. Description des sommets du maillage. Matrice $N \times 3$ avec N nombre de triangles du maillage. Cette matrice contient les numéros de sommets du maillage, triangle par triangle : $T(i, 1 : 3)$ est le vecteur ligne qui contient les trois numéros des sommets du triangle n° i . La numérotation locale des sommets sur le triangle est celle définie par l'ordre des sommets dans $T(i, 1 : 3)$: localement $T(i, 1 : 3)(j)$ est le sommet numéro j du triangle i .
X	Double array. Vecteur ligne. Coordonnées des sommets du maillage. $X(i)$ est la coordonnée en x du sommet n° i .
Y	Double array. Vecteur ligne. Coordonnées des sommets du maillage. $Y(i)$ est la coordonnée en y du sommet n° i .

Tableau 2.2.2-1.

Et les variables du fichier relatif aux EF et aux formules d'intégration :

5.2. APPENDICE. VARIABLES DU CODE : VARIABLES EF (INDICATIVES)

Structure de données du fichier de données relatives aux EF calculées par la fonction EF_Prep	
Ce fichier stocke les variables ci-dessous. L'utilisation du code ne nécessite pas de connaissances sur la signification de ces variables. Comme précédemment on pose $D = \text{donnees.D}$. On note N_E le nombre d'éléments de $D.E$ (cf. Tableau 2.2.1-1c), $F_1, \dots, F_{N_E}$ désignent les différents type d'espaces d'EF choisis parmi ceux définis au Paragraphe 1.4, F_i étant l'espace numéro $D.E(i)$. On note $\{u_j^i\}_{1 \leq j \leq r_i}$ la base de F_i, r_i sa dimension et r_i^{loc} la dimension de l'espace $\mathcal{P}_{m_i, n_i}(K)$ correspondant, pour K triangle du maillage. Ci-dessous N_T, N_A et N_F désignent respectivement pour le maillage, le nombre de triangles, d'arêtes du maillage et d'arêtes sur la frontière de Ω. On suppose que la frontière est partitionnée en n_f parties connexes.	
Données relatives aux matrices d'assemblage	
Ass_Dom	Cell array. Matrice $N_T \times N_E \times 3$. Ass_Dom $\{i, j, k\}$: Double array. Tableau des valeurs, aux points de la formule de quadrature utilisée sur le triangle K de numéro i , de la dérivée d'ordre $k - 1$ des fonctions de base consécutives de l'espace $\mathcal{P}_{m_j, n_j}(K)$ associé à F_j . Le nombre de lignes de Ass_Dom $\{i, j, k\}$ est égal au nombre de points d'intégration et le nombre de colonnes à r_j^{loc} la dimension de $\mathcal{P}_{m_j, n_j}(K)$.
Ass_Ad	Cell array. Matrice $N_A \times N_E \times 3 \times 2$. On note K_1 le triangle numéro Pa ($i, 1$) auquel appartient l'arête i et si cette arête est commune à deux triangles on note K_2 le triangle numéro Pa ($i, 3$) (cf. Tableau 2.2.3-1b, variable Pa). Ass_Ad $\{i, j, k, 1\}$: Double array. Tableau des valeurs, aux points de la formule de quadrature utilisée sur les arêtes de K_1 , de la dérivée d'ordre $k - 1$ des fonctions de base consécutives de l'espace $\mathcal{P}_{m_j, n_j}(K_1)$ associé à F_j . Ass_Ad $\{i, j, k, 2\}$: <i>Idem</i> que le cas précédent si Pa ($i, 3$) est non nul, mais relativement à K_2 . On a Ass_Ad $\{i, j, k, 2\} = []$ si Pa ($i, 3$) est nul (cas d'une arête frontière).
Ass_Fr	Cell array. Matrice $N_F \times N_E \times 3$. Ass_Fr $\{i, j, k\}$: Double array. Tableau des valeurs, aux points de la formule de quadrature utilisée sur les arêtes du triangle K de numéro Pf ($i, 1$) qui porte l'arête numéro i (dans la numérotation locale des arêtes sur la frontière), de la dérivée d'ordre $k - 1$ des fonctions de base consécutives de l'espace $\mathcal{P}_{m_j, n_j}(K)$ associé à F_j (cf. Tableau 2.2.3-1b, variable Pf).

Tableau 2.2.3-1a (Partie 1)

5. APPENDICE

Nalloc_Dom	Double array. Matrice $N_E \times N_E$. Pour la préallocation mémoire de matrices globales d'assemblage. Le terme $\text{Nalloc\_Dom}(i, j)$ est le nombre de termes <i>a priori</i> non nuls de la matrice d'assemblage de la forme bilinéaire $\int_{\Omega} uv$ définie sur $F_j \times F_i$ de terme (p, q) définie par $\int_{\Omega} u_q^j u_p^i$ dont la dimension est $r_i \times r_j$.
Nalloc_Ad	Double array. Matrice $N_E \times N_E$. Equivalent de Nalloc_Dom pour les arêtes du maillage et la forme bilinéaire $\int_{\Gamma} uv + \frac{1}{2} \int_{\Sigma_1} uv + \frac{1}{2} \int_{\Sigma_2} uv$, avec Σ_k , $k = 1$ ou 2 réunion des arêtes intérieures, $k = 1$ signifiant que, sur une arête i donnée, on considère la restriction de uv au triangle de numéro $\text{Pa}(i, 1)$ (<i>cf.</i> Tableau 2.2.3-1b, variable Pa) et $k = 2$ au triangle de numéro $\text{Pa}(i, 3)$.
Nalloc_Fr	Double array. Matrice $N_E \times N_E$. Equivalent pour les termes frontières de Nalloc_Dom avec la forme définie sur la frontière $\int_{\Gamma} uv$.
SL_Dom	Cell array. Matrice $N_E \times N_E \times 3 \times 3$. $\text{SL\_Dom}\{i, j, k, m\}$ est la matrice d'assemblage de la forme bilinéaire définie sur $F_j \times F_i$ par $\int_{\Omega} v_{k-1} w_{m-1}$ et de taille $\dim(F_i) \times \dim(F_j)$. Le terme (p, q) de cette matrice est défini par $\int_{\Omega} u_{q,k-1}^j u_{p,m-1}^i$.
SL_Ad	Cell array. Matrice $N_E \times N_E \times 3 \times 3$. Equivalent de SL_Dom pour les arêtes du domaine et la forme bilinéaire $\int_{\Gamma} uv + \frac{1}{2} \int_{\Sigma_1} uv + \frac{1}{2} \int_{\Sigma_2} uv$.
SL_Fr	Cell array. Matrice $N_E \times N_E \times 3 \times 3$. Equivalent pour la frontière de SL_Ad avec la forme $\int_{\Gamma} uv$.

Tableau 2.2.3-1a (Partie 2 et fin).

5.2. APPENDICE. VARIABLES DU CODE : VARIABLES GEOMETRIE (INDICATIVES)

Suite. Données relatives aux degrés de liberté.	
Pa	Double array. Matrice de taille $N_A \times 5$. Pointeur sur les arêtes numérotées du maillage. Utilisé dans l'assemblage sur les arêtes. $\text{Pa}(j, 1)$ est le numéro de l'un des deux triangles ou du seul triangle du maillage auquel appartient l'arête numéro j . $\text{Pa}(j, 2)$ est le numéro local d'arête sur le triangle numéro $\text{Pa}(j, 1)$ et vaut 1, 2 ou 3. S'il y a une second triangle en commun, $\text{Pa}(j, 3)$ est le numéro de ce second triangle éventuel et $\text{Pa}(j, 4)$ le numéro local d'arête, sinon ces deux termes valent 0. $\text{Pa}(j, 5)$ est le numéro de frontière auquel appartient l'arête j ou vaut 0 lorsqu'elle appartient au domaine. Si on a posé Front =[], alors $\text{Pa}(:, 5)$ est mis à 0 et Pf et Pfc ci-dessous sont vides.
Pf	Double array. Matrice de taille $N_F \times 4$. Pointeur sur les arêtes appartenant à la frontière. $\text{Pf}(i, 1)$ est le numéro du triangle auquel appartient l'arête i , $\text{Pf}(i, 2)$ son numéro local d'arête sur le triangle et vaut 1, 2 ou 3, $\text{Pf}(i, 3)$ son numéro global d'arête et $\text{Pf}(i, 4)$ le numéro de frontière à laquelle elle appartient. Cette variable est utilisée pour l'assemblage sur les arêtes de la frontière. Pf et Pfc ci-dessous donnent la même information compte-tenu du fait que l'on a la relation dans le code : $\text{Pf}(:, 1 : 3) = [\text{Pfc}\{1\}; \dots; \text{Pfc}\{\text{end}\}]$.
Pfc	Cell array. Tableau $1 \times n_f$. Pointeur sur les arêtes de la frontière. Les arêtes de chaque frontière ont aussi une numérotation propre. $\text{Pfc}\{j\}$ est un pointeur relatif a la frontière numéro j . Pour l'arête i de la frontière j , $\text{Pfc}\{j\}(i, 1)$ est le numéro du triangle auquel elle appartient, $\text{Pfc}\{j\}(i, 2)$ est son numéro local d'arête sur le triangle (1,2 ou 3) et $\text{Pfc}\{j\}(i, 3)$ son numéro global d'arête. Utilisation de Pfc dans le code : cf. Arc ci-dessous.
Tc	Cell array. Tableau $1 \times N_E$. $\text{Tc}\{i\}$ est un tableau de dimension $n_T \times r_i^{loc}$ contenant triangle par triangle les numéros des <i>ddl</i> de l'espace F_i .
Xc	Cell array. Tableau $1 \times N_E$. $\text{Xc}\{i\}$ est un vecteur $1 \times r_i$ de doubles. $\text{Xc}\{i\}(j)$ représente les coordonnées en x du degré de liberté numéro j de l'espace F_i .
Yc	Cell array. Tableau $1 \times N_E$. Comme Xc , relativement aux coordonnées en y .
Arc	Cell array. Tableau $1 \times N_E$. $\text{Arc}\{i\}$ est un tableau de N_A lignes tel que $\text{Arc}\{i\}(j, :)$ est le vecteur des numéros de <i>ddl</i> de l'espace F_i placés sur l'arête i . Pour les EF discontinus les <i>ddl</i> sont tous considérés à l'intérieur des triangles et on pose $\text{Arc}\{i\}(j, :) = 0$. Pour les EF continus, les <i>ddl</i> sont parcourus dans l'ordre suivant : d'abord les deux extrémités d'arête (l'ordre est donné par Pa , cf. ci-dessous), puis les <i>ddl</i> obtenus en partant du premier sommet vers le second. Comme la variable Pfc ci-dessus, Arc n'est utilisée que pour les conditions aux limites dans cb , changt_base_nt , changt_base_a et changt_base_nt_t , sans prise en compte de l'orientation d'arête (de l'ordre du premier et second sommet dans $\text{Arc}\{i\}(j, :)$).
Ed	Double array. Vecteur ligne dont le terme i vaut r_i la dimension de F_i .

Tableau 2.2.3-1b

5. APPENDICE

5.3. Paramètres des fonctions décrivant les EDP et le second membre, récapitulatif de tableaux.

**Syntaxe des fonctions sur lesquelles pointent les handles de fonctions
des Tableaux 2.2.1-1b et 2.2.1-1d.**

Syntaxe des fonctions pour les applications a_Ω . Champ feq_Dom.

```
function [Mc, Nc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, c2, ...  
                           de, Ac, Bc, numK) ;
```

Syntaxe des fonctions pour les applications a_Γ . Champ feq_Fr.

```
function [Mc, Nc] = eq_Fr(donnees, numF, uc, a1, b1, c1, ...  
                           a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

Syntaxe des fonctions pour les applications $a_{\bar{\Sigma}}$. Champ feq_Ad.

```
function [Mc, Nc] = eq_Ad(donnees, numF, uc, a1, b1, c1, ...  
                           a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

Syntaxe des fonctions pour les applications b_Ω . Champ fscm_Dom.

```
function [Mc, Nc, Fc] = scm_Dom (donnees, uc, a1, b1, c1, ...  
                                a2, b2, c2, de, Ac, Bc, numK) ;
```

Syntaxe des fonctions pour les applications b_Γ . Champ fscm_Fr.

```
function [Mc, Nc, Fc] = scm_Fr(donnees, numF, uc, a1, b1, c1,...  
                                a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

Syntaxe des fonctions pour les applications $b_{\bar{\Sigma}}$. Champ fscm_Ad.

```
function [Mc, Nc, Fc] = scm_Ad(donnees, numF, uc, a1, b1, c1,...  
                                a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

*Syntaxe des fonctions pour les conditions au bord (voir Paragraphe 5.4 suivant).
Champ fcb.*

```
function [fb,sb] = cb_don(donnees) ;
```

Syntaxe des fonctions pour la méthode de Lesaint-Raviart. Champ D_LR.

```
function [Mc, Nc] = eq_LR(donnees, numF, uc, a1, b1, c1, a2, ...  
                           b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
```

5.3. APPENDICE. PARAMETRES DES FONCTIONS

Fonctions décrivant les équations relatives à $a_D, b_D, D = \Omega, \bar{\Sigma}$ et Γ	
Variables d'entrées communes	
donnees	Variable de type struct. C'est la variable qui décrit les données du problème EDP et dont les champs sont décrits aux tableaux 2.2.1-1a, b, c, d. Cette variable peut contenir d'autre champs définis par l'utilisateur.
uc	Cell array. Tableau $n \times 3$ de cellules. Pour $i = 1, \dots, n$ (n nombres d'inconnues) et $j = 1, 2, 3$, $uc\{i, j\}$ est un vecteur ligne de type double array. Pour $j = 0, 1, 2$, le terme $uc\{i, j+1\}$ représente les valeurs, aux points de la formule de quadrature utilisée pour l'intégration sur le triangle courant (<i>cf.</i> ci-après) ou ses arêtes, de la dérivée d'ordre $j = 0, 1, 2$ de la i -ème composante de l'itéré courant X^0 de (1.5) et (1.10). Ces valeurs sont données par $X_{i,j}^0(\mathbf{Ac}, \mathbf{Bc})$.
a1, b1, c1, a2, b2, c2	Double array. Représentent les coordonnées en x puis en y des trois sommets a, b, c du triangle courant noté K . Le triangle courant K est le triangle du maillage dont le numéro est donné par la variable numK décrite ci-dessous.
de	Double array. Vaut $(b_1 - a_1)(c_2 - a_2) - (c_1 - a_1)(b_2 - a_2)$.
Ac, Bc	Double array. Vecteurs lignes de type double array. Représentent les coordonnées en x et en y des points d'intégration pour l'intégration sur le triangle K ou sur ses arêtes.
numK	Double array. Numéro du triangle courant.
Variable d'entrées pour $D = \Gamma, D = \bar{\Sigma}$ ou la méthode de Lesaint-Raviart.	
numA	Double array. Scalaire désignant le numéro de l'arête courante. La variable numK désigne alors, dans le cas où cette arête est sur la frontière, le numéro du triangle qui porte l'arête et, dans le cas où cette arête est à l'intérieur du domaine, le numéro de l'un des deux triangles qui porte l'arête.
numF	Double array. Scalaire. La frontière peut être partitionnée en plusieurs frontières, numF indique le numéro de frontière auquel appartient l'arête de numéro global numA avec la valeur 0 pour une arête intérieure comme, par exemple, dans la méthode de Lesaint-Raviart. Cette variable se place juste après la variable donnees .
numALoc	Double array. Scalaire désignant le numéro local sur le triangle numK de l'arête courante de numéro global numA .

Tableau 2.2.4-1. Première partie.

5. APPENDICE

Variables de sorties	
$a_D : \text{Mc}, \text{Nc}$	Mc et Nc sont des tableaux de cellules de taille $3 \times 3 \times n_D$ ou bien de taille $3 \times 1 \times n_D$ pour la forme factorisée, avec n_D nombre de pages et $D = \Omega, \bar{\Sigma}, \Gamma$ le type de domaine sur lequel est posée l'équation. Dans le cas de tableaux $3 \times 3 \times n_D$ $\text{Mc}\{i, j, k\}$ et $\text{Nc}\{i, j, k\}$ sont des tableaux $n_{ij}^{D,k} \times n$ de cellules. Dans ce cas les cellules contiennent un vecteur ligne de doubles et $\text{Mc}\{i + 1, j + 1, k\}\{p, q\}(1, m)$ et $\text{Nc}\{i + 1, j + 1, k\}\{p, q\}(1, m)$ représentent les valeurs au point x_m de coordonnées $\text{Ac}(m)$, $\text{Bc}(m)$ de la composante (p, q) des matrices respectives (cf. (1.7)) : $M_{ij}^{D,k}(x_m, X^0(x_m)) \text{ et } N_{ij}^{D,k}(x_m, X^0(x_m)).$ Dans le cas de tableaux $1 \times 3 \times n_D$ qui indiquent au code la forme factorisée, $\text{Mc}\{1, i, k\}$ et $\text{Nc}\{1, i, k\}$ sont alors des tableaux $n^{D,k} \times n$ (taille constante pour une page k donnée). Les cellules contiennent aussi un vecteur ligne de doubles et $\text{Mc}\{1, 1 + i, k\}\{p, q\}(1, m)$ et $\text{Nc}\{1, 1 + i, k\}\{p, q\}(1, m)$ représentent respectivement la valeur de la composante (p, q) de la matrice (cf. (1.9)) : $M_i^{D,k}(x_m, X^0(x_m)) \text{ et } N_i^{D,k}(x_m, X^0(x_m)).$
$b_D : \text{Mc}, \text{Nc}, \text{Fc}$	Mc et Nc sont du même type que ci-dessus. On note n'_D le nombre de pages. On peut de plus considérer des seconds membres ayant m_D^k (fixe pour une page de D donné) composantes au lieu de n qui est la dimension de l'espace EF produit E. Alors dans le cas non factorisé $\text{Mc}\{i, j, k\}$ sera choisie de taille $n_{ij}^{D,k} \times m_D^k$ et dans le cas factorisé $\text{Mc}\{1, i, k\}$ sera choisi de taille $n^{D,k} \times m_D^k$ (cf. (1.7)' et (1.9)'). Fc est un tableau $1 \times 3 \times n_D$ de cellules, telle que $\text{Fc}\{1, i, k\}$ est un vecteur colonne de cellules de taille m_D^k dont chaque cellule contient un vecteur ligne de doubles. $\text{Fc}\{1, 1 + i, k\}\{p, 1\}(1, m)$ représente la valeur de la composante p du second membre $\text{F}_i^{D,k}(x_m, X^0(x_m))$ aux points de coordonnées données par $\text{Ac}(m)$, $\text{Bc}(m)$ (cf. (1.13)).
Mc, Nc	Cas de la méthode de Lesaint-Raviart. Dans ce cas Mc et Nc sont des vecteurs lignes de type double représentant les valeurs aux points de coordonnées données par Ac et Bc de chaque composante du champ suivant lequel on dérive.

Tableau 2.2.4-1. 2ème partie.

5.4. APPENDICE. PARAMETRES DES FONCTIONS POUR LES CONDITIONS AU BORD

5.4. Paramètres des fonctions décrivant les conditions au bord

Syntaxe des fonctions pour les conditions au bord. Champ **fc**.

```
function [fb,sb] = cb_don(donnees) ;
```

Les variables de sortie de cette fonction sont décrites dans les 3 tableaux qui suivent. Dans ces tableaux, le champ principalement utilisé de la variables **sb** est le champ **sb.v** qui permet de définir les fonctions donnant les conditions au bord.

5. APPENDICE

Fonctions décrivant les conditions aux limites	
Variables d'entrée	
donnees	Variable de type struct . Variable décrivant les données du problème.
Variables de sortie	
fb	Vecteur ligne de cellules. La frontière étant partitionnée en n_f parties de frontières connexes. La cellule $\mathbf{fb}\{i\}$ est vecteur ligne de double indiquant les numéros de frontière sur laquelle on a une condition du Dirichlet pour la variable numéro i . On a $\mathbf{fb}\{i\} = []$ s'il n'y a pas de condition de Dirichlet. Un test est réalisé dans cb pour vérifier que fb et $\mathbf{fb}\{i\}$ sont des vecteurs lignes. Ci-après, dans la partie sb.un on notera aussi $\mathbf{fb}\{i\}$ l'ensemble des points formé des composantes de $\mathbf{fb}\{i\}$. Le terme $\mathbf{fb}\{i\}$ définit l'ensemble J_i de <i>ddl</i> entrant dans la définition de l'espace $\mathcal{E}$ des fonctions <i>tests</i> dans (1.2).
sb	Variable de type struct de champs v et un décrits ci-après.
sb.v	Le champs sb.v est un tableau de cellules de taille $n \times n_f$ tels que la cellule $\mathbf{sb.v}\{i, j\}$ est un handle sur la fonction g de deux variables qui définit la condition de Dirichlet pour la variable i sur la frontière j . On pose $\mathbf{sb.v}\{i, j\} = []$ s'il n'y a pas de condition de Dirichlet. Remarque : sb.v peut aussi être choisi de taille $q \times n_f$, avec $q < n$, s'il n'y a pas de conditions de Dirichlet sur les variables de numéro supérieur à $q + 1$. Une variante du code permet de considérer des fonctions g à 3 variables, pour la prise en compte de paramètres, $\mathbf{sb.v}\{i, j\}$ contiendra alors une seconde cellule supplémentaire indiquant le nombre d'argument, exemple $\mathbf{sb.v}\{i, j\} = \{ @g, 3 \}$.
sb.cbc	Conditions au bord : traitement des conditions au bord dans les coins. Les n_f morceaux de frontière qui partitionnent la frontière ont, deux à deux, 0, 1 ou 2 sommets communs. La variable sb.cbc permet de préciser quelle est la condition choisie sur ces sommets. Considérons l'inconnue numéro i . Si sb.cbc ou $\mathbf{sb.cbc}\{i\}$ n'est pas définie, alors si $j_1 < j_2$ sont tels que les frontières de numéro $f_1 = \mathbf{fb}\{i\}(j_1)$ et $f_2 = \mathbf{fb}\{i\}(j_2)$ ont au moins un sommet commun, c'est la condition sur la frontière f_1 qui est prise en compte. Sinon, $\mathbf{sb.cbc}\{i\}$ doit être une matrice de taille $n_f \times n_f$, et la condition au(x) sommet(s) commun(s) est une pondération des conditions sur f_1 et f_2 donné par $\mathbf{sb.cbc}\{i\}(f_1, f_2)$ et $\mathbf{sb.cbc}\{i\}(f_2, f_1)$. Par exemple on posera $\mathbf{sb.cbc}\{i\}(f_1, f_2) = 1$ et $\mathbf{sb.cbc}\{i\}(f_2, f_1) = 0$, pour que ce soit la condition sur f_1 qui soit prise en compte.

Tableau 2.2.4-2a

Fonctions décrivant les conditions aux limites-suite.	
Variables de sorties	
sb.a	Conditions de la forme $AX = U_0$.
Le champ sb.a est soit un tableau de cellules à 3 ou 4 colonnes, soit vaut $\square$, soit n'est pas défini. Chaque ligne de sb.a donne une condition de type $AX = U_0$. Pour i fixé, sb.a$\{i, 1\}$ est un vecteur (ligne ou colonne) qui désigne les composantes $i_1, i_2, \dots, i_p$ de l'inconnue X sur lesquelles portent la condition de type $AX = U_0$. Ces composantes doivent toutes appartenir à un espace d'EF $\tilde{E}$ d'un type commun. La cellule suivante sb.a$\{i, 2\}$ est le vecteur ligne des numéros de morceaux de frontière sur lesquelles portent les conditions, la troisième cellule sb.a$\{i, 3\}$ est le tableau de type double array donné par la matrice B avec $b_{km} = a_{i_k i_m}$ qui est aussi obtenue comme matrice extraite de A en prenant « l'intersection des lignes et colonnes » numéros $i_1, i_2, \dots, i_p$ de A. La matrice B doit être inversible. Les conditions aux limites sont imposées après avoir effectué sur le système un changement de base. Le changement de base porte sur les <i>ddl</i> situées sur les numéros de frontière de sb.a$\{i, 2\}$, ces <i>ddl</i> se rapportant aux inconnues dont les numéros sont donnés par sb.a$\{i, 1\}$. La quatrième cellule est une matrice de taille $n_f \times n_f$ (n_f nombre de morceaux de frontière). Elle permet de définir une priorité dans le choix du changement de base, pour des <i>ddl</i> situés à l'intersection de deux morceaux de frontières pour lesquelles peuvent figurer deux changements de base distincts. Soient $i_1, i_2, \dots, i_p$ les indices de sb.a$\{i, 1\}$. On pose $u = (X_{i_1}, X_{i_2}, \dots, X_{i_p})$. Le terme sb.a$\{i, 2\}$ doit être un vecteur formé de termes de $\mathbf{fb}\{i_1\} \cup \mathbf{fb}\{i_2\} \cup \mathbf{fb}\{i_p\}$. Soit $j \in \mathbf{sb.a}\{i, 2\}$, alors si $j \in \mathbf{fb}\{i_1\}$, il y a donc une condition imposée sur X_{i_1} sur la frontière numéro j, celle-ci sera remplacée par la condition $a_{i_1 i_1} u_{i_1} + a_{i_1 i_2} u_{i_2} + \dots + a_{i_1 i_p} u_{i_p} = u_{0i_1}$. La valeur de u_{0i_1} sur ce morceau de frontière est alors donné par la fonction de handle sb.v$\{i_1, j\}$. On raisonne ensuite de la même manière pour les conditions imposées sur $X_{i_2}, \dots, X_{i_p}$. Les espaces d'EF $E_{i_1}, E_{i_2}, \dots, E_{i_p}$ de l'espace produit E doivent être des espaces identiques. Un test de compatibilité de $E_{i_1}, E_{i_2}, \dots, E_{i_p}$ est réalisé. Le terme sb.a$\{i, 4\}$ est construit de la manière suivante, sb.a$\{i, 4\}$ est la matrice nulle $n_f \times n_f$ excepté pour les termes suivants. Si un <i>ddl</i> appartient à deux frontières numérotées f_1 et f_2, avec f_1 composantes de sb.a$\{i, 2\}$, alors le changement de base choisi sera celui de la frontière f_1 si on pose sb.a$\{i, 4\}(f_1, f_2) = 1$, celui de la frontière f_2 si sb.a$\{i, 4\}(f_2, f_1) = 1$ et f_2 composantes de sb.a$\{i, 2\}$ et aucun changement de base sinon. Les lignes i de sb.a permettent de traiter d'autres couples distincts d'inconnues (dans la version actuelle les couples doivent en principe être tous distincts) ou pour un couple donné, plusieurs listes de numéros de frontières. Si la cellule sb.a$\{i, 4\}$ n'existe pas, les conditions au bord sont imposées dans l'ordre des lignes de sb.a et sur chaque ligne i dans l'ordre des numéros de sb.a$\{i, 2\}$, à l'intersection de deux frontières : c'est alors la dernière condition rencontrée qui prime.	

Tableau 2.2.4-2b. Exemples au Paragraphe 2.6.3.

Fonctions décrivant les conditions aux limites-suite.	
Variables de sorties	
sb.un	Conditions normales et tangentes sur un champ de vitesse formé à partir des inconnues.
Les conditions de Dirichlet imposées sur un (ou plusieurs) champ de vecteurs (u_1, u_2) choisi deviennent des conditions sur $(u.n, u.t)$ (n normale et t tangente). Le champ sb.un est soit non défini ou vide, soit une matrice de type cell de 2 à 4 colonnes et dans ce cas, chaque ligne de sb.un traite le cas d'un vecteur u. Sur chaque ligne, la première cellule est un vecteur ligne ou colonne de deux termes qui désigne les deux composantes de l'inconnue qui définissent le champ u. La seconde cellule est le vecteur ligne de numéros de morceaux de frontière sur lesquelles la condition de Dirichlet sur u_1 (resp. u_2) devient une condition de Dirichlet sur $u.n$ (resp. $u.t$). La troisième cellule est soit facultative, soit vide, soit est une valeur numérique qui correspond à un angle de rotation, cette angle est nul lorsque la cellule est vide ou non définie. La quatrième cellule est une matrice de taille $n_f \times n_f$ (n_f nombre de morceaux de frontière) relative au choix d'une normale à l'intersection de deux morceaux de frontière (comme par exemple dans le cas de coins). Soit $i_1 = \mathbf{sb.un}\{i, 1\}(1)$ et $i_2 = \mathbf{sb.un}\{i, 1\}(2)$, on pose $u = (X_{i_1}, X_{i_2})$. Le terme $\mathbf{sb.un}\{i, 2\}$ doit être un vecteur formé de termes de $\mathbf{fb}\{i_1\} \cup \mathbf{fb}\{i_2\}$. Soit $j \in \mathbf{sb.un}\{i, 2\}$, alors si $j \in \mathbf{fb}\{i_1\}$, il y a donc une condition imposée sur X_{i_1} sur la frontière numéro j, celle-ci sera remplacée par une condition sur $u.n$ avec n normale extérieure unitaire. La valeur de $u.n$ sur ce morceau de frontière est alors donné par la fonction de handle $\mathbf{sb.v}\{i_1, j\}$. De même si $j \in \mathbf{fb}\{i_2\}$, il y a donc aussi une condition imposée sur X_{i_2} sur la frontière numéro j, celle-ci sera remplacée par une condition sur $u.t$, avec t vecteur unitaire tangent tel que la base (n, t) soit directe. La valeur de $u.t$ sur ce morceau de frontière est alors donnée par $\mathbf{sb.v}\{i_2, j\}$. Comme pour sb.a, un changement de base est effectuée sur le système les inconnues relatives $\mathbf{sb.un}\{i, 1\}$ et à leur <i>ddl</i> sur les frontières numéro $\mathbf{sb.un}\{i, 2\}$, avec comme nouvelle base, la base (n, t). Les espaces E_{i_1}, E_{i_2} termes de l'espace EF produit E doivent être des espaces identiques. Un test est réalisé dans chngt_base_nt pour vérifier que $\mathbf{sb.un}\{i\}(2)$ est un vecteur ligne, ainsi qu'un test de compatibilité de E_{i_1} et E_{i_2}. Lorsque $\mathbf{sb.un}\{i, 3\}$ existe, est non vide et de valeur θ, alors les conditions sont imposées sur $u.n'$ et $u.t'$ au lieu de $u.n$ et $u.t$, (n', t') étant la base obtenue par rotation de θ de la base (n, t). Lorsque $\mathbf{sb.un}\{i, 4\}$ existe et est non vide, soit $A = \mathbf{sb.un}\{i, 4\}$ qui doit être alors une matrice $n_f \times n_f$. Si un sommet de la frontière appartient à deux frontières numérotées f_1 et f_2 composantes de $\mathbf{sb.un}\{i, 2\}$, alors la normale choisie est une pondération de la normale sur f_1 avec le poids $A(f_1, f_2)$ et de la normale sur f_2 avec le poids $A(f_2, f_1)$. Les lignes i de sb.un permettent de traiter d'autres couples distincts d'inconnues (dans la version actuelle les couples doivent en principe être tous distincts) ou pour un couple donné, plusieurs listes de numéros de frontières.	

Tableau 2.2.4-2c. Exemples au Paragraphe 2.10.7.

5.5. Modèle de fonction pour les conditions au bord

Les exemples de fichiers qui suivent ci-dessous sont utilisés pour le problème de Stokes ($\mathcal{P}_6^{(1)}$) posé sur le carré unité $[0, 1] \times [0, 1]$, au Paragraphe 2.10 :

$$(\mathcal{P}_6^{(1)}) \left\{ \begin{array}{l} \text{Il existe } (u, p) \text{ tel que :} \\ -\eta \Delta u + \nabla p = 0 \text{ sur } \Omega, \\ \nabla \cdot u = 0 \text{ sur } \Omega, \\ u(x, y) = 0 \text{ sur } \Gamma_1 \cup \Gamma_3 \text{ (parois } y = 0 \text{ et } y = 1), \\ u(x, y) = (4y(1 - y), 0) \text{ sur } \Gamma_4 \cup \Gamma_2 \text{ (entrée } x = 0 \text{ et sortie } x = 1). \end{array} \right.$$

Ce problème est résolu, avec $\eta = 1$, par la commande

```
cd C :/EF_text/exemples/exemple_stokes, instruct1 ;
```

Les fichiers exemples de fonctions qui suivent et qui résolvent ce problème sont mis en œuvre par la commande

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

avec une utilisation de formes factorisées pour décrire les EDP (*cf.* (1.8), (1.12)) et, à titre d'exemple, car elle n'intervient pas ici, la méthode de Lesant-Raviart (*cf.* (1.17)), ou bien par

```
cd C :/EF_text/exemples/modeles, instruct2 ;
```

avec une utilisation de formes non factorisées pour décrire les EDP sur le domaine (*cf.* (1.6), (1.11)).

Ce problème est aussi résolu, comme au Paragraphe 2.10.7, sur le carré obtenu par rotation de $\frac{\pi}{3}$ du carré ci-dessus, par les commandes

```
cd C :/EF_text/exemples/modeles, instruct1_rotation ;
```

pour les formes factorisées ou bien par

```
cd C :/EF_text/exemples/modeles, instruct2_rotation ;
```

avec utilisation de formes non factorisées,

Nous traitons d'abord le cas des conditions au bord. Les 3 inconnues du problème de Stokes sont notées successivement u_1 et u_2 (vitesse du fluide) et p (pression) que l'on notera aussi u_3 (troisième inconnue). On désigne par n la normale unitaire à la frontière et t le vecteur tangent tels que (n, t) est une base orthonormale directe. Dans cet exemple on impose les conditions sur $u = (u_1, u_2)$ aux frontières 1 et 3 et à titre d'exemple sur $u = (u_1, u_2)$ aux frontières 2 et 4 mais en imposant les conditions cette fois sur le vecteur $v = (v_1, v_2)$ de composantes définies par $v_1 = (u_1, u_2) \cdot n$ et $v_2 = (u_1, u_2) \cdot t$.

La variable **fb** indique les numéros d'inconnues et de frontière concernés : on pose

```
fb{1} = [1 :4] ;
fb{2} = [1 :4] ;
```

fb{3} = [] ;

Ensuite, sur les morceaux frontières de 1 et 3, on impose la condition nulle au bord pour u_1 et u_2 en posant $\mathbf{sb.v}\{1, i\} = \mathbf{sb.v}\{2, i\} = \mathbf{@f1}$ pour $i = 1, 3$.

Sur les morceaux frontières de 2 et 4, on impose sur v_1 et v_2 le profil de vitesse, pour cela on pose $\mathbf{sb.v}\{1, i\} = \mathbf{@f2}$ (Poiseuille), et $\mathbf{sb.v}\{2, i\} = \mathbf{@f1}$ (condition nulle au bord) pour $i = 2, 4$.

Pour que les conditions soient effectivement imposées sur (v_1, v_2) , on pose :

- $\mathbf{sb.un}\{1\} = [1, 2]$: le vecteur $[1, 2]$ désigne les numéros des deux inconnues (u_1, u_2) concernées.

- $\mathbf{sb.un}\{1, 2\} = [2, 4]$: le vecteur $[2, 4]$ désigne les frontières 2 et 4 concernées.

Ici, les conditions au bord sur les frontières désignées par $\mathbf{sb.un}\{1, 2\} = [2, 4]$, au lieu de porter sur le vecteur $u = (u_1, u_2)$ (les indices sont désignés par $\mathbf{sb.un}\{1, 2\} = [1, 2]$) portent alors, respectivement, sur les composantes normales et tangentielle de ce vecteur.

Il est à noter que chaque ligne de **sb.un** concerne un vecteur donné.

Nous donnons ci-dessous (page suivante) le fichier des conditions au bord, on pourra utiliser ce fichier sans les commentaires détaillés qu'il comporte, laissés ici à titre explicatif.

Fichier *EF_text/exemples/modeles/equ/cb_don.m*

```
function [fb,sb] = cb_don(donnees)
%-----%
%CB_DON Fonction decrivant les conditions au bord (CB). On note
%  U = (U1,U2,U3) l'inconnue sur laquelle porte les conditions au
%  bord, avec u = (U1,U2) la vitesse et p = U3 la pression.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%
%VARIABLES DE SORTIE :
%-----%
%fb : CELL ARRAY. Vecteur ligne de N termes ou N est le nombre
%  d'inconnues.
%  fb{I} DOUBLE ARRAY. Vecteur ligne d'entiers compris entre 1 et
%  NF ou NF est le nombre de morceaux de frontieres, ou bien
%  fb{I}= []. Chaque entier composante du vecteur fb{I} designe
%  un numero de frontiere sur laquelle est imposee une condition
%  aux limites portant sur l'inconnue numero I.
%
%sb : STRUCT ARRAY.
%  sb.v : CELL ARRAY de taille N x NF.
%  sb.v{I,J} : FUNCTION_HANDLE ARRAY. Handle sur la fonction de
%  deux variables qui definit sur la frontiere numero fb{I}(J)
%  la condition aux limites portant sur la variable numero I.
%
%---PRISE EN COMPTE DES CONDITIONS AU BORD SUR LES COMPOSANTES---%
%-----NORMALES ET TANGENTIELLES DE LA VITESSE-----%
%
%sb.un : CELL ARRAY de 2 a 4 colonnes, ou bien sb.un = [], ou bien
%  sb.un n'est pas defini (ce champ est facultatif). Chaque ligne
```

```

% de sb.un traite le cas d'un vecteur forme de deux composantes
% de l'inconnue U. Relativement a l'utilisation de ce champ, on
% note n la normale unitaire exterieure a la frontiere et t la
% tangente telle que (n,t) soit une base directe de  $R^2$ .
%
% sb.un{I,1} : DOUBLE ARRAY. Vecteur ligne de deux termes. On
% pose :
%         I1 = sb.un{I,1}(1)
%         et I2 = sb.un{I,1}(2).
%
%sb.un{I,2} : DOUBLE ARRAY. Vecteur forme des termes de
%         fb{I1} U fb{I2}.
%
% Soit VI = (UI1,UI2) le champ de vecteur, sur la fronti-
% ere numero J, J appartenant a sb.un{I,2} on a alors :
%
%         - au lieu de porter sur UI1 la condition au bord (fonction
%         sb.v{I1,J}) porte sur VI.n.
%
%         - au lieu de porter sur UI2 la condition au bord (fonction
%         sb.v{I2,J}) porte sur VI.t.
%
%-----DEBUT DE LA FONCTION-----%
%Frontieres sur lesquelles on a une condition au bord :
fb{1} = [1 :4] ;
fb{2} = [1 :4] ;
fb{3} = [] ;
%Valeur des conditions au bord.
%Initialisation a la fonction nulle :
for i = [1,3]
    sb.v{1,i} = @f1 ;
    sb.v{2,i} = @f1 ;
end ;
%Vitesse du fluide a l'entree du domaine :

```

5.5. APPENDICE. MODELES DE FONCTIONS : CONDITIONS AU BORD. (*cb_don.m*)

```

sb.v{1,4} = @f2 ;
sb.v{2,4} = @f1 ;
%Vitesse du fluide a la sortie du domaine :
sb.v{1,2} = @f3 ;
sb.v{2,2} = @f1 ;

%-----CONDITIONS SUR (U1,U2).n et (U1,U2).T-----%
%Changement de condition par sb.v.
%  On pose des conditions sur v1 = (u1,u2).n et v2 = (u1,u2).t
%  au lieu de u1 et u2 : pour cela on pose sb.un{1,1} = [1,2].
%  Ces conditions ont lieu sur les frontieres 2 et 4 : pour cela
%  on pose sb.un{1,2} =[2,4] :
sb.un{1} = [1,2] ;
sb.un{1,2} = [2,4] ;

%Matrice pour des CB du type Au = B. Cette matrice est ici donnee
%  a titre d'exemple car elle est facultative dans le cadre de ce
%  probleme. Ici, on pose A matrice unite de R^2. Cf doc.
%
sb.a{1,1} = [1,2] ;
sb.a{1,2} = [1,2,3,4] ;
sb.a{1,3} = [1,0;0,1] ;

%Matrice de priorite aux intersections des frontieres. Facultatif
%  ici, car il n'y a pas de discontinuite des CB en ces points.
%  Ici sb.cbc est tel que :
%  - la frontiere numero 1 est prioritaire sur les frontieres
%  adjacentes numeros 2 et 4.
%  - la frontiere numero 3 est prioritaire sur les frontieres
%  adjacentes numeros 2 et 4. Cf. doc.
sb.cbc{1} = [0,1,0,1 ; 0,0,0,0 ; 0,1,0,1 ; 0,0,0,0] ;

%-----FIN DE LA FONCTION-----%

```

```
%-----SOUS-FONCTIONS-----%  
%Fonction nulle de  $\mathbb{R}^2$  sur  $\mathbb{R}$  :  
function z = f1(x,y)  
z = 0.*x ;  
%Fonction entree du fluide :  
function z = f2(x,y) ;  
z = 4*y.*(y - 1) ;  
%Fonction sortie du fluide :  
function z = f3(x,y) ;  
z = -4*y.*(y - 1) ;
```


5.6. MODELE DE FONCTION DECRIVANT L'EDP SUR LE DOMAINE.

5.6. Modèle de fonction décrivant les EDP sur le domaine (forme factorisée)

On reprend le problème de Stokes du Paragraphe 2.10. La fonction ci-dessous décrit les matrices du Paragraphe 2.10.2 qui définissent le problème EDP sur le domaine avec une forme factorisée. Le choix de matrices est le suivant :

$$M_0 = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}; \quad M_1 = 2\eta \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ -\frac{1}{2\eta} & 0 & 0 \end{pmatrix}; \quad M_2 = 2\eta \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{2\eta} & 0 \end{pmatrix};$$

et

$$N_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad N_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

qui correspond à l'assemblage de la forme bilinéaire :

$$\int_{\Omega} (M_0 u + M_1 u_{,1} + M_2 u_{,2}, N_0 u + N_1 u_{,1} + N_2 u_{,2}).$$

Dans le fichier qui suit, on a rajouté à titre d'exemple, une deuxième page de matrices. Cette page ne modifie pas les équations car les matrices choisies dans cette page sont toutes nulles. Ces matrices sont données par :

$$M_0 = M_1 = M_2 = N_0 = N_1 = N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Cette fonction est mise en œuvre lors de l'exécution de la commande :

```
cd C :/EF_text/exemples/modeles, instruct1;
```

Le fichier de fonctions décrivant l'EDP est donné à la page suivante.

Fichier EF_text/exemples/modeles/equ/eq_Dom.m

```
function [Mc, Nc] = eq_Dom(donnees, uc, a1, b1, c1, a2, b2, ...
                           c2, de, Ac, Bc, numK);
%-----%
%EQ_DOM. Fonction decrivant l'EDP sur le domaine.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%  On remarque que de est un produit vectoriel et que l'on a
%  abs(de) = 2*aire(K).
```

5.6. MODELE DE FONCTION DECRIVANT L'EDP SUR LE DOMAINE.

```
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
% points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
% de sommets sont donnees par T(numK, :), T etant le tableau des
% sommets de maillage dans la structure de maillage adoptee dans
% le code.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Mc{I,J,K} : CELL ARRAY.
% Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
% de taille MIJK x N avec N nombre d'inconnues.
% Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
% de taille MK x N. Dans ce cas le nombre de ligne est constant
% pour chaque page et ne depend pas de J.
% Mc{I,J,K}{P,Q} ou Mc{I,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. Pour la forme factorisee la taille est
% 1 x 3 x P.
% Nc{I,J,K} : CELL ARRAY.
% Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3, Nc{1,I,K} est un tableau de cellules de
% taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{I,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
```

```

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
%   Mc{1,J,K} et Nc{1,J,K} sont MK x N avec N nombre d'inconnues %
%
%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
eta = 1;
D = 2*eta.*Un;

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
%   Mc{1,J,1} qui designe aussi Mc{J} : de taille M1 x N = 5 x 3,
%   Nc{1,J,1} qui designe aussi Nc{J} : de taille N1 x N = 5 x 3.
%   N est le nombre d'inconnues et on a necessairement N1 = M1
%   (compatibilite produit matriciel).
%
Mc{1} = {
    Zero, Zero, -Un;
    Zero, Zero, Zero;
    Zero, Zero, Zero;
    Zero, Zero, -Un;
    Zero, Zero, Zero
};
Mc{2} = {
    D, Zero, Zero;
    Zero, D/2, Zero;
    Zero, D/2, Zero;
    Zero, Zero, Zero;
    -Un, Zero, Zero
};

```

5.6. MODELE DE FONCTION DECRIVANT L'EDP SUR LE DOMAINE.

```

};
Mc{3} = {
    Zero, Zero, Zero ;
    D/2, Zero, Zero ;
    D/2, Zero, Zero ;
    Zero, D, Zero ;
    Zero, -Un, Zero
};
Nc{1} = {
    Zero, Zero, Zero ;
    Zero, Zero, Zero ;
    Zero, Zero, Zero ;
    Zero, Zero, Zero ;
    Zero, Zero, Un
};
Nc{2} = {
    Un, Zero, Zero ;
    Zero, Un/2, Zero ;
    Zero, Un/2, Zero ;
    Zero, Zero, Zero ;
    Zero, Zero, Zero ;
};
Nc{3} = {
    Zero, Zero, Zero ;
    Un/2, Zero, Zero ;
    Un/2, Zero, Zero ;
    Zero, Un, Zero ;
    Zero, Zero, Zero ;
};
%-----PAGE 2-----%
% Mc{1,J,2} de taille M2 x N = 3 x 3,
% Nc{1,J,2} de taille N2 x N = 3 x 3.
% On a necessairement M2 = N2, M2 et N2 peuvent differer de M1,
% N1 (cf. premiere page).

```

```

Mc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
%-----FIN DE LA FONCTION-----%
```

5.7. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE SUR LE DOMAINE.

5.7. Modèle de fonction décrivant le second membre sur le domaine (forme factorisée)

Pour le second membre, les matrices du Paragraphe 2.10.2 sont les suivantes

$$\tilde{M}_0 = \tilde{N}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}; \quad \tilde{M}_1 = \tilde{M}_2 = \tilde{N}_1 = \tilde{N}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$F_0 = \begin{pmatrix} f_1(x) \\ f_2(x) \\ 0 \end{pmatrix}, \quad F_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad F_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

qui correspond à l'assemblage de la forme linéaire :

$$\int_{\Omega} (\tilde{M}_0 F_0 + \tilde{M}_1 F_1 + \tilde{M}_2 F_2, \tilde{N}_0 v + \tilde{N}_1 v_{,1} + \tilde{N}_2 v_{,2}).$$

Dans le fichier qui suit, on a posé $f_1 = f_2 = 0$.

On a rajouté à titre d'exemple, une deuxième page de matrices et de second membre. Cette page ne modifie par les équations car les matrices choisies sont toutes nulles. Les matrices ont ici 4 lignes et les matrices de second membre sont de taille 5 au lieu de 3 (pour chaque page, on peut choisir pour le nombre de composantes du second membre, une autre taille que le nombre d'inconnues). Ces matrices sont données par :

$$\tilde{M}_0 = \tilde{M}_1 = \tilde{M}_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad \tilde{N}_0 = \tilde{N}_1 = \tilde{N}_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$F_0(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad F_1(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad F_2(x) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Commande de mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct1;
```

Le fichier de fonctions décrivant le second membre de l'EDP sur le domaine est donné à la page suivante.

Fichier EF_text/exemples/modeles/equ/scm_Dom.m

```
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, a2, b2,...
                                c2, de, Ac, Bc, numK) ;

%-----%
%SCM_DOM. Fonction decrivant le second membre de l'EDP sur le
% domaine.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
% structure de donnees qui definit le probleme EF.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
% ou le chiffre 3 correspond aux trois derivations.
% Represente les valeurs de l'itere courant et de ses derivees
% aux points de la formule de quadrature pour l'integration sur
% le triangle courant note ici K. Les coordonnees de ces points
% d'integration sont donnees par les vecteurs Ac et Bc.
% uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
% Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
% partielle d'ordre DER, avec DER = 0,1,2, aux points de la
% formule de quadrature de l'inconnue numero I.
%-----%
% a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
% du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
% la bijection affine du triangle de reference de sommets (0,0),
% (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
% On remarque que de est un produit vectoriel et que l'on a
```


5.7. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE SUR LE DOMAINE.

```
%   abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%   points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%   de sommets sont donnees par T(numK, :), T etant le tableau des
%   sommets de maillage dans la structure de maillage adoptee dans
%   le code.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%   le nombre de pages. La taille 1 x 3 x P correspond a la forme
%   factorisee.
%   Mc{I,J,K} : CELL ARRAY.
%   Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
%   de taille MIJK x NFK avec NFK taille des vecteurs colonnes
%   Fc{1,1,K}, Fc{1,2,K}, Fc{1,3,K}.
%   Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
%   de taille MK x NFK. Dans ce cas le nombre de ligne est
%   constant pour chaque page et ne depend pas de J.
%   Mc{I,J,K}{P,Q} ou Mc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
%   de length(Ac) termes.
%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%   le nombre de pages. La taille 1 x 3 x P correspond a la forme
%   factorisee.
%   Nc{I,J,K} : CELL ARRAY.
%   Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
%   de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
%   Lorsque Nc est 1 x 3 x P, Nc{1,J,K} est un tableau de cellules
%   de taille NK x N avec NK = MK (nombre de ligne constant par
%   page).
```

```

% Nc{I,J,K}{P,Q} ou Nc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%
%Fc : CELL ARRAY. Tableau 1 x 3 x P de cellules.
% Fc{1,J,K} : CELL ARRAY. Vecteur colonne de taille NFK.
% Fc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne de length(Ac)
% termes.
%-----%

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
%-----Fc est de taille 1 x 3 x P-----%
% Fc{1,J,K} est NFK x 1, NFK a choisir suivant le probleme
% Mc{1,J,K} est MK x NFK, MK a choisir suivant le probleme
% Nc{1,J,K} est MK x N, N nombre d'inconnues

%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
% Second membre de taille 3
% Fc{1,J,1}, qui designe aussi Fc{J}, de taille NF1 x 1 = 3 x 1
% Mc{1,J,1}, qui designe aussi Mc{J}, de taille M1 x NF1 = 3 x 3
% Nc{1,J,1}, qui designe aussi Nc{J}, de taille N1 x N = 3 x 3
% (On doit avoir N1 = M1, on a N = 3)
%
Mc{1,1,1} = {Un, Zero, Zero;
             Zero, Un, Zero;
             Zero, Zero, Zero;
             };

```

5.7. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE SUR LE DOMAINE.

```

Mc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };

Mc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };

Nc{1,1,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };

Nc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };

Nc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };

%-----PAGE 2-----%
% Second membre de taille 5
% Fc{1,J,2} est de taille NF2 x 1 = 5 x 1
% Mc{1,J,2} est de taille M2 x NF2 = 4 x 5
% Nc{1,J,2} est de taille N2 x N = 4 x 3 (N2 = M2 = 4, N = 3)
%
Mc{1,1,2} = {Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              };

Mc{1,2,2} = {Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;

```

```

                Zero, Zero, Zero, Zero, Zero ;
                                } ;
Mc{1,3,2} = {Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
              Zero, Zero, Zero, Zero, Zero ;
                                } ;
Nc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
                                } ;
Nc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
                                } ;
Nc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
                                } ;

%-----SECOND MEMBRE-----%
%Initialisation de Fc :
%EXEMPLE :
%-----PAGE 1-----%
% Fc{1,I,1} de taille NF1 x 1 = 3 x 1
Fc{1} = {f1(Ac,Bc) ;
          f2(Ac,Bc) ;
          Zero} ;
Fc{2} = {Zero ;
          Zero ;
          Zero} ;

```

5.7. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE SUR LE DOMAINE.

```
Fc{3} = {Zero ;  
        Zero ;  
        Zero} ;  
  
%-----PAGE 2-----%  
%   Fc{1,I,2} de taille NF2 x 1 = 5 x 1  
Fc{1,1,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;  
Fc{1,2,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;  
Fc{1,3,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;  
%  
%-----FIN DE LA FONCTION-----%  
  
%-----SOUS-FONCTIONS-----%  
%EXEMPLES de sous-fonctions :  
function z = f1(x,y) ;  
z = 0*x.^0 ;  
  
function z = f2(x,y) ;  
z = 0*x.^0 ;
```

5.8. Modèle de fonction décrivant les EDP sur la frontière (forme factorisée)

La contribution de la frontière est nulle. A titre d'exemple, on pose pour la première page :

$$M_0 = M_1 = M_2 = N_0 = N_1 = N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

qui correspond à l'assemblage de la forme bilinéaire :

$$\int_{\Omega} (M_0 u + M_1 u_{,1} + M_2 u_{,2}, N_0 v + N_1 v_{,1} + N_2 v_{,2}).$$

Ensuite, pour la seconde page on pose :

$$M_0 = M_1 = M_2 = N_0 = N_1 = N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Commande de mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

Le fichier de fonctions décrivant l'EDP sur la frontière est donné à la page suivante.

Par rapport à la fonction du Paragraphe 5.6 (EDP sur le domaine), la syntaxe des variables d'entrée est différente avec la variable d'entrée supplémentaire **numF** le numéro de frontière courante, la variable **numA** qui est le numéro, dans la numérotation locale des arêtes sur la frontière, de l'arête courante, et la variable **numALoc** qui est le numéro local de l'arête courante, en tant qu'arête sur le triangle de numéro **numK**.

5.8. MODELE DE FONCTION DECRIVANT L'EDP : TERMES FRONTIERES.

Fichier EF_text/exemples/modeles/equ/eq_Fr.m

```
function [Mc, Nc] = eq_Fr(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%EQ_FR. Fonction decrivant l'EDP sur la frontiere.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
```

```

%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%      On remarque que de est un produit vectoriel et que l'on a
%      abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
% points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
% de sommets sont donnees par T(numK, :), T etant le tableau des
% sommets de maillage dans la structure de maillage adoptee dans
% le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
% donnees relatif aux espaces d'EF et aux formules de
% quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
% triangle numero numK.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Mc{I,J,K} : CELL ARRAY.
% Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
% de taille MIJK x N avec N nombre d'inconnues.
% Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
% de taille MK x N. Dans ce cas le nombre de ligne est constant
% pour chaque page et ne depend pas de J.
% Mc{I,J,K}{P,Q} ou Mc{I,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%

```


5.8. MODELE DE FONCTION DECRIVANT L'EDP : TERMES FRONTIERES.

```
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. Pour la fome factorisee la taille est
% 1 x 3 x P.
% Nc{I,J,K} : CELL ARRAY.
% Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3, Nc{1,I,K} est un tableau de cellules de
% taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{I,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
% Mc{1,J,K} et Nc{1,J,K} sont MK x N avec N nombre d'inconnues %
%
%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
% Mc{1,J,1} qui designe aussi Mc{J} : de taille M1 x N = 3 x 3
% Nc{1,J,1} qui designe aussi Nc{J} : de taille N1 x N = 3 x 3
% (M1 = N1)
Mc{1,1,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
```

```

};
Mc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,1,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
%-----PAGE 2-----%
%   Mc{1,J,2} de taille M2 x N = 4 x N   (N = 3)
%   Nc{1,J,2} de taille N2 x N = 4 x N   (N2 = M2)
%
Mc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;

```

5.8. MODELE DE FONCTION DECRIVANT L'EDP : TERMES FRONTIERES.

```

                Zero, Zero, Zero ;
};
Nc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
%-----FIN DE LA FONCTION-----%
```

5.9. Modèle de fonction décrivant le second membre sur la frontière (forme factorisée)

Cette fonction est construite sur le même modèle que la fonction du Paragraphe 5.7 (second membre sur le domaine) avec un choix identique des matrices. Seule la syntaxe des variables d'entrée est différente.

Commande de mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

Le fichier de fonctions décrivant le second membre de l'EDP sur la frontière est donné à la page suivante.

5.9. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE : TERMES FRONTIERES.

Fichier EF_text/exemples/modeles/equ/scm_Fr.m

```
function [Mc, Nc, Fc] = scm_Fr(donnees, numF, uc, a1, b1, c1, ...
                             a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%SCM_FR. Fonction decrivant le second membre sur la frontiere.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
```

```

%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%      On remarque que de est un produit vectoriel et que l'on a
%      abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
% points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
% de sommets sont donnees par T(numK, :), T etant le tableau des
% sommets de maillage dans la structure de maillage adoptee dans
% le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
% donnees relatif aux espaces d'EF et aux formules de
% quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
% triangle numero numK.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Mc{I,J,K} : CELL ARRAY.
% Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
% de taille MIJK x NFK avec NFK taille des vecteurs colonnes
% Fc{1,1,K}, Fc{1,2,K}, Fc{1,3,K}.
% Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
% de taille MK x NFK. Dans ce cas le nombre de ligne est
% constant pour chaque page et ne depend pas de J.
% Mc{I,J,K}{P,Q} ou Mc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.

```

5.9. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE : TERMES FRONTIERES.

```
%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Nc{I,J,K} : CELL ARRAY.
% Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3 x P, Nc{1,J,K} est un tableau de cellules
% de taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%

%Fc : CELL ARRAY. Tableau 1 x 3 x P de cellules.
% Fc{1,J,K} : CELL ARRAY. Vecteur colonne de taille NFK.
% Fc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne de length(Ac)
% termes.
%-----%

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
%-----Fc est de taille 1 x 3 x P-----%
% Fc{1,J,K} est NFK x 1, NFK a choisir suivant le probleme
% Mc{1,J,K} est MK x NFK, MK a choisir suivant le probleme
% Nc{1,J,K} est MK x N, N nombre d'inconnues
%
%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
%
%-----PAGE 1-----%
```

Second membre de taille 3

Fc{1,J,1}, qui designe aussi Fc{J}, de taille NF1 x 1 = 3 x 1

```

%  Mc{1,J,1}, qui designe aussi Mc{J}, de taille M1 x NF1 = 3 x 3
%  Nc{1,J,1}, qui designe aussi Nc{J}, de taille N1 x N  = 3 x 3
%  (On doit avoir N1 = M1, on a  N = 3)
%
Mc{1,1,1} = {Un,   Zero, Zero ;
              Zero, Un,   Zero ;
              Zero, Zero, Zero ;
              };
Mc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };
Mc{1,3,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,1,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,2,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,3,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
%-----PAGE 2-----%
%  Second membre de taille 5
%  Fc{1,J,2} est de taille NF2 x 1 = 5 x 1
%  Mc{1,J,2} est de taille M2 x NF2 = 4 x 5
%  Nc{1,J,2} est de taille N2 x N  = 4 x 3 (N2 = M2 = 4, N = 3)
%
```


5.9. MODELE DE FONCTION DECRIVANT LE SECOND MEMBRE : TERMES FRONTIERES.

```
Mc{1,1,2} = {Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              } ;
```

```
Mc{1,2,2} = {Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              } ;
```

```
Mc{1,3,2} = {Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              Zero,  Zero, Zero, Zero, Zero ;
              } ;
```

```
Nc{1,1,2} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              } ;
```

```
Nc{1,2,2} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              } ;
```

```
Nc{1,3,2} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              } ;
```

```
%-----SECOND MEMBRE-----%
```

```
%Initialisation de Fc :
```

```
%EXEMPLE :
```

```

%-----PAGE 1-----%
%   Fc{1,I,1} de taille NF1 x 1 = 3 x 1
Fc{1} = {f1(Ac,Bc) ;
          f2(Ac,Bc) ;
          Zero} ;
Fc{2} = {Zero ;
          Zero ;
          Zero} ;
Fc{3} = {Zero ;
          Zero ;
          Zero} ;

%-----PAGE 2-----%
%   Fc{1,I,2} de taille NF2 x 1 = 5 x 1
Fc{1,1,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,2,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,3,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
%
%-----FIN DE LA FONCTION-----%

%-----SOUS-FONCTIONS-----%
%EXEMPLES de sous-fonctions :
function z = f1(x,y) ;
z = 0*x.^0 ;

function z = f2(x,y) ;
z = 0*x.^0 ;

```

5.10. Modèle de fonction décrivant les EDP sur les arêtes du maillage (forme factorisée)

La contribution des arêtes du maillage (hormis celle due aux conditions de type Dirichlet vue au Paragraphe 5.5) est nulle. A titre d'exemple, on pose, comme pour le cas de la frontière au Paragraphe 5.8, pour la première page :

$$M_0 = M_1 = M_2 = N_0 = N_1 = N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

qui correspond à l'assemblage de la forme bilinéaire :

$$\int_{\Omega} (M_0 u + M_1 u_{,1} + M_2 u_{,2}, N_0 v + N_1 v_{,1} + N_2 v_{,2}).$$

Ensuite pour la seconde page on pose :

$$M_0 = M_1 = M_2 = N_0 = N_1 = N_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Commande de mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

Le fichier de fonctions décrivant l'EDP sur les arêtes du maillage est donné à la page suivante.

Par rapport à la fonction du Paragraphe 5.6 (EDP sur le domaine), la syntaxe des variables d'entrée est différente avec la variable d'entrée supplémentaire **numF** le numéro de frontière courante, la variable **numA** qui est le numéro, dans la numérotation globale des arêtes du maillage, de l'arête courante, et la variable **numALoc** qui est le numéro local de l'arête courante, en tant qu'arête sur le triangle de numéro **numK**.

Fichier EF_text/exemples/modeles/equ/eq_Ad.m

```
function [Mc, Nc] = eq_Ad(donnees, numF, uc, a1, b1, c1, a2, ...
                        b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%EQ_AD. Fonction decrivant l'EDP sur les aretes.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
```

5.10. MODELE DE FONCTION DECRIVANT L'EDP : ARETES DU MAILLAGE.

```
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%      On remarque que de est un produit vectoriel et que l'on a
%      abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%      points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%      de sommets sont donnees par T(numK, :), T etant le tableau des
%      sommets de maillage dans la structure de maillage adoptee dans
%      le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
%      donnees relatif aux espaces d'EF et aux formules de
%      quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
%      triangle numero numK.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%      le nombre de pages. Pour la fome factorisee la taille est
%      1 x 3 x P.
%      Mc{I,J,K} : CELL ARRAY.
%      Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
%      de taille MIJK x N avec N nombre d'inconnues.
%      Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
%      de taille MK x N. Dans ce cas le nombre de ligne est constant
%      pour chaque page et ne depend pas de J.
%      Mc{I,J,K}{P,Q} ou Mc{I,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
%      de length(Ac) termes.
%-----%
```

```

%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Nc{I,J,K} : CELL ARRAY.
% Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3, Nc{1,I,K} est un tableau de cellules de
% taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{I,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
% Mc{1,J,K} et Nc{1,J,K} sont MK x N avec N nombre d'inconnues %
%
%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
% Mc{1,J,1} qui designe aussi Mc{J} : de taille M1 x N = 3 x 3
% Nc{1,J,1} qui designe aussi Nc{J} : de taille N1 x N = 3 x 3
% (M1 = N1)
Mc{1,1,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;

```

5.10. MODELE DE FONCTION DECRIVANT L'EDP : ARETES DU MAILLAGE.

```

};
Mc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,1,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Nc{1,3,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
%-----PAGE 2-----%
%   Mc{1,J,2} de taille M2 x N = 4 x N   (N = 3)
%   Nc{1,J,2} de taille N2 x N = 4 x N   (N2 = M2)
%
Mc{1,1,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,2,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
};
Mc{1,3,2} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;

```

```

        Zero, Zero, Zero ;
};
Nc{1,1,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
};
Nc{1,2,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
};
Nc{1,3,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
};
%-----FIN DE LA FONCTION-----%
```


5.11. MODELE DE FONCTION DECRIVANT LE SCD MEMBRE : ARETES DU MAILLAGE.

5.11. Modèle de fonction décrivant le second membre sur les arêtes du maillage (forme factorisée)

Cette fonction est construite sur le même modèle que la fonction du Paragraphe 5.7 (second membre sur le domaine) avec un choix identique des matrices. Seule la syntaxe des variables d'entrée est différente.

Commande de mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

Le fichier de fonctions décrivant le second membre de l'EDP sur les arêtes du maillage est donné à la page suivante.

Fichier EF_text/exemples/modeles/equ/scm_Ad.m

```
function [Mc, Nc, Fc] = scm_Ad(donnees, numF, uc, a1, b1, c1, ...
                               a2, b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%SCM_AD. Fonction decrivant le second membre sur les aretes.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
```

5.11. MODELE DE FONCTION DECRIVANT LE SCD MEMBRE : ARETES DU MAILLAGE.

```
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%      On remarque que de est un produit vectoriel et que l'on a
%      abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%      points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%      de sommets sont donnees par T(numK, :), T etant le tableau des
%      sommets de maillage dans la structure de maillage adoptee dans
%      le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
%      donnees relatif aux espaces d'EF et aux formules de
%      quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
%      triangle numero numK.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%      le nombre de pages. La taille 1 x 3 x P correspond a la forme
%      factorisee.
%      Mc{I,J,K} : CELL ARRAY.
%      Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
%      de taille MIJK x NFK avec NFK taille des vecteurs colonnes
%      Fc{1,1,K}, Fc{1,2,K}, Fc{1,3,K}.
%      Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
%      de taille MK x NFK. Dans ce cas le nombre de ligne est
%      constant pour chaque page et ne depend pas de J.
%      Mc{I,J,K}{P,Q} ou Mc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
%      de length(Ac) termes.
```

```

%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
% le nombre de pages. La taille 1 x 3 x P correspond a la forme
% factorisee.
% Nc{I,J,K} : CELL ARRAY.
% Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3 x P, Nc{1,J,K} est un tableau de cellules
% de taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%

%Fc : CELL ARRAY. Tableau 1 x 3 x P de cellules.
% Fc{1,J,K} : CELL ARRAY. Vecteur colonne de taille NFK.
% Fc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne de length(Ac)
% termes.
%-----%

%-----DEBUT DE LA FONCTION-----%
%-----FORME FACTORISEE : Mc et Nc sont 1 x 3 x P-----%
%-----Fc est de taille 1 x 3 x P-----%
% Fc{1,J,K} est NFK x 1, NFK a choisir suivant le probleme
% Mc{1,J,K} est MK x NFK, MK a choisir suivant le probleme
% Nc{1,J,K} est MK x N, N nombre d'inconnues
%
%Variables intermediaires :
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
%
%-----PAGE 1-----%
% Second membre de taille 3
% Fc{1,J,1}, qui designe aussi Fc{J}, de taille NF1 x 1 = 3 x 1

```

5.11. MODELE DE FONCTION DECRIVANT LE SCD MEMBRE : ARETES DU MAILLAGE.

```

%   Mc{1,J,1}, qui designe aussi Mc{J}, de taille M1 x NF1 = 3 x 3
%   Nc{1,J,1}, qui designe aussi Nc{J}, de taille N1 x N  = 3 x 3
%   (On doit avoir N1 = M1, on a  N = 3)
%
Mc{1,1,1} = {Un,   Zero, Zero ;
              Zero, Un,   Zero ;
              Zero, Zero, Zero ;
              };
Mc{1,2,1} = {Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              Zero, Zero, Zero ;
              };
Mc{1,3,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,1,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,2,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };
Nc{1,3,1} = {Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              Zero,  Zero, Zero ;
              };

%-----PAGE 2-----%
%   Second membre de taille 5
%   Fc{1,J,2} est de taille NF2 x 1 = 5 x 1
%   Mc{1,J,2} est de taille M2 x NF2 = 4 x 5
%   Nc{1,J,2} est de taille N2 x N  = 4 x 3 (N2 = M2 = 4, N = 3)
%
```

```
Mc{1,1,2} = {Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             } ;
```

```
Mc{1,2,2} = {Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             } ;
```

```
Mc{1,3,2} = {Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             Zero, Zero, Zero, Zero, Zero ;
             } ;
```

```
Nc{1,1,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             } ;
```

```
Nc{1,2,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             } ;
```

```
Nc{1,3,2} = {Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             Zero, Zero, Zero ;
             } ;
```

```
%-----SECOND MEMBRE-----%
```

```
%Initialisation de Fc :
```

```
%EXEMPLE :
```

5.11. MODELE DE FONCTION DECRIVANT LE SCD MEMBRE : ARETES DU MAILLAGE.

```
%-----PAGE 1-----%
%   Fc{1,I,1} de taille NF1 x 1 = 3 x 1
Fc{1} = {f1(Ac,Bc) ;
          f2(Ac,Bc) ;
          Zero} ;
Fc{2} = {Zero ;
          Zero ;
          Zero} ;
Fc{3} = {Zero ;
          Zero ;
          Zero} ;

%-----PAGE 2-----%
%   Fc{1,I,2} de taille NF2 x 1 = 5 x 1
Fc{1,1,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,2,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,3,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
%
%-----FIN DE LA FONCTION-----%

%-----SOUS-FONCTIONS-----%
%EXEMPLES de sous-fonctions :
function z = f1(x,y) ;
z = 0*x.^0 ;

function z = f2(x,y) ;
z = 0*x.^0 ;
```

5.12. Modèles de fonctions pour la méthode de Lesaint-Raviart

Voici le *script* des deux fonctions appelées lors de l'instruction

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

pour la résolution du problème de Stokes ($\mathcal{P}_6^{(1)}$) vu au Paragraphe 5.5.

Le champ suivant lequel on dérive défini par chacune ces deux fonctions est nul dans les deux cas (ici la méthode Lesaint-Raviart ne modifie pas le problème EF, elle est utilisée à titre d'exemple).

Un exemple d'initialisation du champ `D.LR` de la variable `donnees` de type `struct` décrivant la structure de données du problème EF pour la prise en compte de ces fonctions est donné au Paragraphe 5.16 dans le fichier *don1.m*.

Dans le fichier *don1.m* les commandes relatives à la méthode de Lesaint-Raviart sont :

```
%Methode de Lesaint-Raviart
fLR1 = @eq_LR1 ;
%Autres fonctions pour Lesaint-Raviart
fLR2 = @eq_LR2 ;
...
%Premiere derivation :
D.LR(1, :) = {fLR1, [1,2]} ;
%Autres derivations :
D.LR(2, :) = {fLR2, [3] } ;
...
%Description des espaces d'EF :
donnees.D = D ;
```

pour une dérivation suivant le champ de vitesse défini par `fLR1` des composantes 1 et 2 et, suivant le champ de vitesses `fLR2`, de la troisième composante. Les fichiers de fonctions (identiques, hormis le nom) sont donnés à la page suivante.

5.12. MODELE DE FONCTION DECRIVANT L'EDP : LESAIN-T-RAVIART.

Fichier EF_text/exemples/modeles/equ/eq_LR1.m

```
function [Mc, Nc] = eq_LR1(donnees, numF, uc, a1, b1, c1, a2, ...
                           b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%EQ_LR1. Champ de vitesse suivant lequel on derive. Methode de
%  Lesaint-Raviart.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
```

```

%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%  On remarque que de est un produit vectoriel et que l'on a
%  abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%  points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%  de sommets sont donnees par T(numK, :), T etant le tableau des
%  sommets de maillage dans la structure de maillage adoptee dans
%  le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
%  donnees relatif aux espaces d'EF et aux formules de
%  quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
%  triangle numero numK.
%
%VARIABLES DE SORTIES
%-----%
%Mc : DOUBLE ARRAY : Vecteur ligne de length(Ac) termes. Premiere
%  composante de la vitesse.
%-----%
%Mc : DOUBLE ARRAY : Vecteur ligne de length(Ac) termes. Deuxieme
%  composante de la vitesse.
%-----%
%Le champ suivant lequel on derive est nul :
Mc = 0*uc{1,1};
Nc = 0*uc{2,1};

```

5.12. MODELE DE FONCTION DECRIVANT L'EDP : LESAIN-T-RAVIART.

Fichier EF_text/exemples/modeles/equ/eq_LR2.m

```
function [Mc, Nc] = eq_LR2(donnees, numF, uc, a1, b1, c1, a2, ...
                           b2, c2, de, Ac, Bc, numK, numALoc, numA) ;
%-----%
%EQ_LR2. Champ de vitesse suivant lequel on derive. Methode de
%  Lesaint-Raviart.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%numF : DOUBLE ARRAY. Numero de frontiere a laquelle appartient
%  l'arete courante de numero numA avec la valeur 0 pour les
%  aretes interieures.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
```

```

%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%  On remarque que de est un produit vectoriel et que l'on a
%  abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%  points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%  de sommets sont donnees par T(numK, :), T etant le tableau des
%  sommets de maillage dans la structure de maillage adoptee dans
%  le code.
%-----%
%numA : DOUBLE ARRAY. Numero global d'arete dans la structure de
%  donnees relatif aux espaces d'EF et aux formules de
%  quadrature : variable Pa.
%-----%
%numAloc : DOUBLE ARRAY. Numero local de l'arete numA sur le
%  triangle numero numK.
%
%VARIABLES DE SORTIES
%-----%
%Mc : DOUBLE ARRAY : Vecteur ligne de length(Ac) termes. Premiere
%  composante de la vitesse.
%-----%
%Mc : DOUBLE ARRAY : Vecteur ligne de length(Ac) termes. Deuxieme
%  composante de la vitesse.
%-----%
%Le champ suivant lequel on derive est nul :
Mc = 0*uc{1,1};
Nc = 0*uc{2,1};

```

5.13. Modèle de fonction décrivant les EDP sur le domaine (forme non factorisée)

On reprend la résolution du Problème de Stokes ($\mathcal{P}_6^{(1)}$) vue au Paragraphe 5.5. On utilise cette fois-ci une forme non factorisée pour décrire l'EDP sur le domaine, au lieu de la forme factorisée utilisée au Paragraphe 5.6. De même, au Paragraphe 5.14 suivant, on utilisera aussi une forme non factorisée pour décrire cette fois le second membre de l'EDP sur le domaine, à la place de la forme factorisée vue au Paragraphe 5.7. La commande de résolution est la suivante :

```
cd C :/EF_text/exemples/modeles, instruct2
```

Le cas de l'utilisation d'une forme factorisée pour les équations sur les frontières et les arêtes du domaine se traite de manière identique.

La fonction exemple donnée ci-dessous décrit les matrices utilisées au Paragraphe 4.3 dans l'Exemple 4 de la description de la fonction *EF_assemblage.m*. Il s'agit dans cet exemple du problème de l'écoulement à l'intérieur d'un carré d'un fluide de Stokes, avec entraînement du fluide sur la face supérieure du carré. Cet exemple est mis en œuvre par la commande :

```
cd C :/EF/exemples, exemple3_EF_assemblage ;
```

La forme bilinéaire assemblée est la suivante :

$$\begin{aligned} & 2\eta(d(u), d(v)) - (p, \nabla \cdot v) - (\nabla \cdot u, q) \\ & = 2\eta \left(u_{1,1}v_{1,1} + \frac{1}{2}(u_{1,2}v_{1,2} + u_{1,2}v_{2,1} + u_{2,1}v_{1,2} + u_{2,1}v_{2,1}) + u_{2,2}v_{2,2} \right) \\ & \quad - pv_{1,1} - pv_{2,2} - u_{1,1}q - u_{2,2}q. \end{aligned}$$

Le fichier exemple utilise une représentation factorisée de cette forme bilinéaire, de la forme

$$\int_{\Omega} \sum_{0 \leq i,j \leq 2} (M_{ij}u_{,i}, N_{ij}v_{,j})$$

avec les matrices suivantes :

$$M_{00} = N_{00} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

et pour les matrices $(M_{ij})_{0 \leq i,j \leq 2}$, $(i,j) \neq (0,0)$ non nulles, dans l'ordre d'initialisation du fichier, avec $D = 2\eta$:

$$\begin{aligned} M_{11} &= \begin{pmatrix} D & 0 & 0 \\ 0 & D/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad M_{21} = \begin{pmatrix} D/2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad M_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & D/2 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \\ M_{22} &= \begin{pmatrix} D/2 & 0 & 0 \\ 0 & D & 0 \\ 0 & 0 & 0 \end{pmatrix}; \quad M_{01} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}; \quad M_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \end{pmatrix}; \end{aligned}$$

$$M_{10} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{20} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

et pour les matrices $(N_{ij})_{0 \leq i,j \leq 2}$,

$$N_{11} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{21} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$N_{22} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{01} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}; N_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix};$$

$$N_{10} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; N_{12} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix};$$

A titre d'exemple ce fichier contient une deuxième page de matrices qui sont :

$$M_{00} = N_{00} = M_{01} = N_{01} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{02} = N_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$M_{10} = N_{10} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{11} = N_{11} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$M_{20} = N_{20} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$M_{21} = N_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; M_{22} = N_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Dans la fonction exemple, pour la seconde page, les matrices sont initialisées par une boucle. Le nombre de lignes des matrices $M_{i-1,j-1}$, $N_{i-1,j-1}$, $1 \leq i, j \leq 2$ est alors donné par le terme (i, j) de la matrice $\mathbf{M}$ suivant initialisée dans la commande qui précède la boucle :

$$\mathbf{M} = \begin{pmatrix} 2 & 2 & 3 \\ 3 & 5 & 2 \\ 3 & 3 & 4 \end{pmatrix}.$$

La mise en œuvre de la fonction décrivant les équations se fait avec la commande

```
cd C :/EF_text/exemples/modeles, instruct2;
```

Le fichier de fonction correspondant à ces deux pages de matrices est alors donné à la page suivante, avec la valeur $\eta = 1$.

5.13. MODELE DE F^n DECRIVANT L'EDP SUR LE DOMAINE (FORME NON FACTORISEE).

Fichier EF_text/exemples/modeles/equ/eq_Dom_nf.m

```
function [Mc, Nc] = eq_Dom_nf(donnees, uc, a1, b1, c1, a2, b2, ...
                             c2, de, Ac, Bc, numK);
%-----%
%EQ_DOM. Fonction decrivant l'EDP sur le domaine.
%  CF utilisation EXEMPLE2b_EF_ASSEMBLAGE
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%  On remarque que de est un produit vectoriel et que l'on a
```

```

%   abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%   points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%   de sommets sont donnees par T(numK, :), T etant le tableau des
%   sommets de maillage dans la structure de maillage adoptee dans
%   le code.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%   le nombre de pages. Pour la forme factorisee la taille est
%   1 x 3 x P.
%   Mc{I,J,K} : CELL ARRAY.
%   Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
%   de taille MIJK x N avec N nombre d'inconnues.
%   Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
%   de taille MK x N. Dans ce cas le nombre de ligne est constant
%   pour chaque page et ne depend pas de J.
%   Mc{I,J,K}{P,Q} ou Mc{I,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
%   de length(Ac) termes.
%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%   le nombre de pages. Pour la forme factorisee la taille est
%   1 x 3 x P.
%   Nc{I,J,K} : CELL ARRAY.
%   Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
%   de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
%   Lorsque Nc est 1 x 3, Nc{1,I,K} est un tableau de cellules de
%   taille NK x N avec NK = MK (nombre de ligne constant par
%   page).
%   Nc{I,J,K}{P,Q} ou Nc{I,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne

```


5.13. MODELE DE F^n DECRIVANT L'EDP SUR LE DOMAINE (FORME NON FACTORISEE).

```
% de length(Ac) termes.

%-----DEBUT DE LA FONCTION-----%
%-----FORME NON FACTORISEE : Mc et Nc sont 3 x 3 x P-----%
% Mc{1,J,K} et Nc{1,J,K} sont MIJK x N avec N nombre d'inconnues %
%
%
%EXEMPLE Variables intermediaires
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
%Viscosite donnee par l'utilisateur
eta = 1;
D = 2*eta.*Un;

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
% Mc{1,J,1} qui designe aussi Mc{J} : de taille 3 x 3 (N = 3)
% Nc{1,J,1} qui designe aussi Nc{J} : de taille 3 x 3 (N = 3)
%
for i=1 :3
    for j=1 :3
        for k=1 :3
            for m=1 :3
                Mc{i,j,1}{k,m} = Zero;
                Nc{i,j,1}{k,m} = Zero;
            end;
        end;
    end;
end;

%Vitesse :
%u1,1*v1,1
Mc{2,2,1}{1,1} = D;
```

$$Nc\{2,2,1\}\{1,1\} = Un ;$$

$$\%1/2*u2,1*v2,1$$

$$Mc\{2,2,1\}\{2,2\} = D/2 ;$$

$$Nc\{2,2,1\}\{2,2\} = Un ;$$

$$\%1/2*u1,2*v2,1$$

$$Mc\{3,2,1\}\{1,1\} = 1/2*D ;$$

$$Nc\{3,2,1\}\{1,2\} = Un ;$$

$$\%1/2*u2,1*v1,2$$

$$Mc\{2,3,1\}\{2,2\} = 1/2*D ;$$

$$Nc\{2,3,1\}\{2,1\} = Un ;$$

$$\%1/2*u1,2*v1,2$$

$$Mc\{3,3,1\}\{1,1\} = D/2 ;$$

$$Nc\{3,3,1\}\{1,1\} = Un ;$$

$$\%u2,2*v2,2$$

$$Mc\{3,3,1\}\{2,2\} = D ;$$

$$Nc\{3,3,1\}\{2,2\} = Un ;$$

$$\%Pression$$

$$\%-p*v1,1$$

$$Mc\{1,2,1\}\{3,3\} = -Un ;$$

$$Nc\{1,2,1\}\{3,1\} = Un ;$$

$$\%-p*v2,2$$

$$Mc\{1,3,1\}\{3,3\} = -Un ;$$

$$Nc\{1,3,1\}\{3,2\} = Un ;$$

$$\%-u1,1*q$$

$$Mc\{2,1,1\}\{1,1\} = -Un ;$$

$$Nc\{2,1,1\}\{1,3\} = Un ;$$

5.13. MODELE DE F^n DECRIVANT L'EDP SUR LE DOMAINE (FORME NON FACTORISEE).

```
%-u2,2*q
Mc{3,1,1}{2,2} = -Un ;
Nc{3,1,1}{2,3} = Un ;
%-----PAGE 2-----%
%EXEMPLE Mc{I,J,2} de taille MIJ2 x N = M(i,j) x 3
%      Nc{I,J,2} de taille NIJ2 x N = M(i,j) x 3 (NIJ2=MIJ2, N=3)
%      M(i,j) est la matrice ci-dessous

M = [2,2,3 ;
      3,5,2 ;
      3,3,4] ;
for i=1 :3
    for j=1 :3
        for k=1 :M(i,j)
            for m=1 :3
                Mc{i,j,2}{k,m} = Zero ;
            end ;
        end ;
    end ;
end ;
for i=1 :3
    for j=1 :3
        for k=1 :M(i,j)
            for m=1 :3
                Nc{i,j,2}{k,m} = Zero ;
            end ;
        end ;
    end ;
end ;
%-----FIN DE LA FONCTION-----%
```

5.14. Modèle de fonction décrivant le second membre sur le domaine (forme non factorisée)

On donne ici l'exemple d'une forme non factorisée pour le second membre. Le cadre est celui du Paragraphe 5.13 précédent. Dans le fichier exemple dont le *script* est donné ci-dessous, la première page de matrices est donnée par

$$\tilde{M}_{ij} = \tilde{N}_{ij} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; 1 \leq i, j \leq 3$$

$$F_0 = \begin{pmatrix} f_1(x) \\ f_2(x) \\ 0 \end{pmatrix}, F_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, F_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

qui correspond à l'assemblage de la forme linéaire :

$$\int_{\Omega} \sum_{0 \leq i, j \leq 2} (\tilde{M}_{ij} F_i, \tilde{N}_{ij} v_{,j}).$$

Dans le fichier exemple, on a posé $f_1 = f_2 = 0$. On a rajouté à titre d'exemple, une deuxième page de matrices et de second membre, avec cette fois-ci un second membre de taille 5 au lieu de 3. Cette page ne modifie pas les équations car les matrices choisies sont toutes nulles. Les matrices ont le même nombre de lignes qu'au paragraphe précédent. On a

$$\tilde{M}_{00} = \tilde{M}_{01} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \tilde{M}_{02} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$\tilde{M}_{10} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \tilde{M}_{11} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$\tilde{M}_{20} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix};$$

$$\tilde{M}_{20} = \tilde{M}_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \tilde{M}_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$\tilde{N}_{00} = \tilde{N}_{01} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \tilde{N}_{02} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

5.14. MODELE DE F^n : SECOND MEMBRE SUR LE DOMAINE (FORME NON FACTORISEE).

$$\tilde{N}_{10} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \tilde{N}_{11} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix};$$

$$\tilde{N}_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\tilde{N}_{20} = \tilde{N}_{21} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}; \tilde{N}_{22} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

et pour le second membre on pose

$$F_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, F_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, F_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Mise en œuvre de cette fonction :

```
cd C :/EF_text/exemples/modeles, instruct2 ;
```

Le fichier de fonction correspondant est alors le suivant (page suivante).

Fichier EF_text/exemples/modeles/equ/scm_Dom_nf.m

```
function [Mc, Nc, Fc] = scm_Dom(donnees, uc, a1, b1, c1, a2, b2,...
                                c2, de, Ac, Bc, numK) ;

%-----%
%SCM_DOM. Fonction decrivant le second membre de l'EDP sur le
%  domaine.
%
%-----%
%SCM_DOM. Fonction decrivant le second membre de l'EDP sur le
%  domaine.
%
%VARIABLES D'ENTREE :
%-----%
%DONNEES : STRUCT ARRAY. Variable de type STRUCT decrivant la
%  structure de donnees qui definit le probleme EF.
%-----%
%uc : CELL ARRAY. MATRICE N x 3 ou N est le nombre d'inconnues et
%  ou le chiffre 3 correspond aux trois derivations.
%  Represente les valeurs de l'itere courant et de ses derivees
%  aux points de la formule de quadrature pour l'integration sur
%  le triangle courant note ici K. Les coordonnees de ces points
%  d'integration sont donnees par les vecteurs Ac et Bc.
%  uc{i,j} : DOUBLE ARRAY. Vecteur ligne de longueur LENGTH(Ac).
%  Le vecteur uc{I,DER + 1} represente les valeurs de la derivee
%  partielle d'ordre DER, avec DER = 0,1,2, aux points de la
%  formule de quadrature de l'inconnue numero I.
%-----%
%  a1, b1, c1, a2, b2, c2 : DOUBLE ARRAY. Coordonnees des sommets
%  du triangle courant K de numero numK.
%-----%
%de : DOUBLE ARRAY. Determinant de la matrice Jacobienne de
%  la bijection affine du triangle de reference de sommets (0,0),
%  (1,0),(0,1) sur le triangle courant K. Sa valeur est donnee par
```

5.14. MODELE DE F^n : SECOND MEMBRE SUR LE DOMAINE (FORME NON FACTORISEE).

```
%
%      de = [a1, b1, c1]*[b2 - c2, -a2 + c2, -b2 + a2]'.
%
%  On remarque que de est un produit vectoriel et que l'on a
%  abs(de) = 2*aire(K).
%-----%
%Ac, Bc : DOUBLE ARRAY. Vecteurs ligne des coordonnees en X, Y des
%  points de la formule de quadrature.
%-----%
%numK : DOUBLE ARRAY. Numero du triangle courant K dont les numeros
%  de sommets sont donnees par T(numK, :), T etant le tableau des
%  sommets de maillage dans la structure de maillage adoptee dans
%  le code.
%
%VARIABLES DE SORTIE
%-----%
%Mc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%  le nombre de pages. La taille 1 x 3 x P correspond a la forme
%  factorisee.
%  Mc{I,J,K} : CELL ARRAY.
%  Lorsque Mc est 3 x 3 x P, Mc{I,J,K} est un tableau de cellules
%  de taille MIJK x NFK avec NFK taille des vecteurs colonnes
%  Fc{1,1,K}, Fc{1,2,K}, Fc{1,3,K}.
%  Lorsque Mc est 1 x 3 x P, Mc{1,J,K} est un tableau de cellules
%  de taille MK x NFK. Dans ce cas le nombre de ligne est
%  constant pour chaque page et ne depend pas de J.
%  Mc{I,J,K}{P,Q} ou Mc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
%  de length(Ac) termes.
%-----%
%Nc : CELL ARRAY. Tableau 3 x 3 x P ou 1 x 3 x P de cellules, P est
%  le nombre de pages. La taille 1 x 3 x P correspond a la forme
%  factorisee.
%  Nc{I,J,K} : CELL ARRAY.
%  Lorsque Nc est 3 x 3 x P, Nc{I,J,K} est un tableau de cellules
```

```

% de taille NIJK x N avec NIJK = MIJK et N nombre d'inconnues.
% Lorsque Nc est 1 x 3 x P, Nc{1,J,K} est un tableau de cellules
% de taille NK x N avec NK = MK (nombre de ligne constant par
% page).
% Nc{I,J,K}{P,Q} ou Nc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne
% de length(Ac) termes.
%-----%
%Fc : CELL ARRAY. Tableau 1 x 3 x P de cellules.
% Fc{1,J,K} : CELL ARRAY. Vecteur colonne de taille NFK.
% Fc{1,J,K}{P,Q} : DOUBLE ARRAY : Vecteur ligne de length(Ac)
% termes.
%-----%

%-----DEBUT DE LA FONCTION-----%
%-----FORME NON FACTORISEE : Mc et Nc sont 3 x 3 x P-----%
%-----Fc est de taille 1 x 3 x P-----%
% Fc{1,J,K} est NFK x 1, NFK a choisir suivant le probleme
% Mc{I,J,K} est MIJK x NFK, MIJK a choisir suivant le probleme
% Nc{I,J,K} est MIJK x N, N nombre d'inconnues

%Variables intermediaires
%EXEMPLE :
Zero = zeros(1,length(Ac));
Un = ones(1,length(Ac));
%Viscosite donnee par l'utilisateur
eta = 1;
D = 2*eta.*Un;

%Initialisation de Mc et Nc :
%EXEMPLE :
%-----PAGE 1-----%
%Second membre de taille 3
% Fc{1,J,1} est de taille NF1 x 1 = 3 x 1
% Mc{I,J,1} de taille MIJ1 x NF1 = 3 x 3

```


5.14. MODELE DE F^n : SECOND MEMBRE SUR LE DOMAINE (FORME NON FACTORISEE).

```
% Nc{I,J,1} de taille NIJ1 x N = 3 x 3 (NIJ1 = MIJ1, N = 3)
%
for i=1 :3
    for j=1 :3
        for k=1 :3
            for m=1 :3
                Mc{i,j,1}{k,m} = Zero ;
                Nc{i,j,1}{k,m} = Zero ;
            end ;
        end ;
    end ;
end ;
%-----PAGE 2-----%
%Second membre de taille 5
% Fc{1,J,2} est de taille NF2 x 1 = 5 x 1
% Mc{I,J,2} de taille MIJ2 x NF2 = M(i,j) x 5
% Nc{I,J,2} de taille NIJ2 x N = M(i,j) x 3 (NIJ2 = MIJ2, N = 3)
% M(i,j) est la matrice ci-dessous
%
M = [2,2,3 ;
      3,5,2 ;
      3,3,4] ;
for i=1 :3
    for j=1 :3
        for k=1 :M(i,j)
            for m=1 :5
                Mc{i,j,2}{k,m} = Zero ;
            end ;
        end ;
    end ;
end ;
for i=1 :3
    for j=1 :3
        for k=1 :M(i,j)
```

```

        for m=1 :3
            Nc{i,j,2}{k,m} = Zero ;
        end ;
    end ;
end ;

%-----SECOND MEMBRE-----%
%Initialisation de Fc :
%EXEMPLE :
%-----PAGE 1-----%
%   Fc{1,I,1} de taille NF1 x 1 = 3 x 1
Fc{1} = {f1(Ac,Bc) ; f2(Ac,Bc) ; Zero} ;
Fc{2} = {Zero ; Zero ; Zero} ;
Fc{3} = {Zero ; Zero ; Zero} ;
%
%-----PAGE 2-----%
%   Fc{1,I,2} de taille NF2 x 1 = 5 x 1
Fc{1,1,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,2,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
Fc{1,3,2} = {Zero ; Zero ; Zero ; Zero ; Zero} ;
%
%-----FIN DE LA FONCTION-----%

%-----SOUS-FONCTIONS-----%
%EXEMPLES de sous-fonctions :
function z = f1(x,y) ;
z = 0*x.^0 ;

function z = f2(x,y) ;
z = 0*x.^0 ;

```

5.15. MODELE DE FICHIER SCRIPT EXECUTANT LE PROGRAMME.

5.15. Modèle de fichier script exécutant le programme

Voici un modèle de fichier *script* pour la résolution du problème de Stokes ($\mathcal{P}_6^{(1)}$) abordé au Paragraphe 5.5. Ce fichier est mis en œuvre par la commande

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

Nous donnons ci-dessous la figure sauvegardée lors de l'exécution de cette commande.

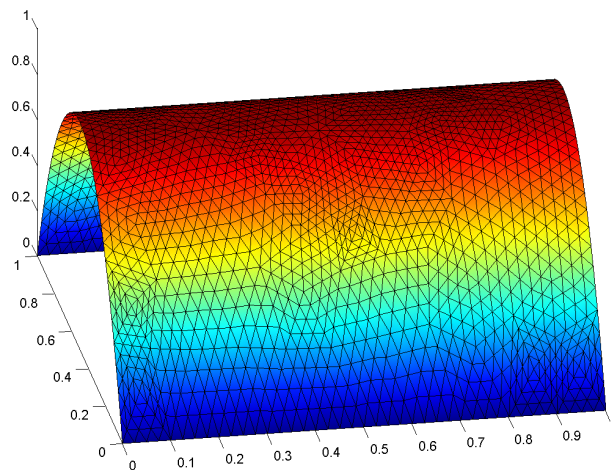


Figure A-1. Problème ($\mathcal{P}_{6h}^{(1)}$) : écoulement de Poiseuille. Graphe de la première composante de la vitesse. Approximation $[P_2^c]^2 \times P_1^c$. La solution exacte est solution du problème approché. Pour le tracé, les triangles du maillage sont subdivisés en quatre triangles égaux.

Nous donnons aussi la figure obtenue par la commande

```
cd C :/EF_text/exemples/modeles, instruct1_rotation ;
```

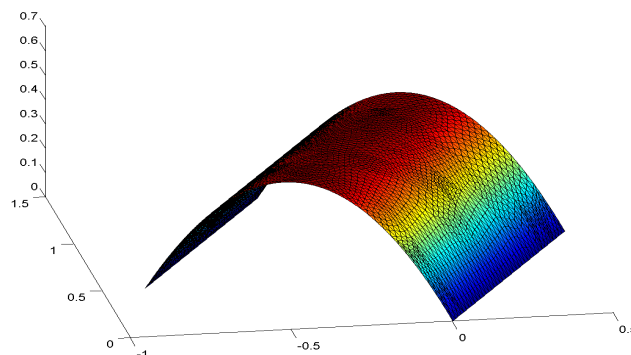


Figure A-1b. Analogie de l'écoulement de Poiseuille de la figure A-1, le carré $[0, 1] \times [0, 1]$ ayant subi ici une rotation de $\frac{\pi}{3}$.

La fonction d'initialisation de données est décrite au Paragraphe 5.16 qui suit. Le fichier *instruct1.m* est le suivant (page suivante).

Fichier EF_text/exemples/modeles/instruct1.m

```
%Fichier d'instructions pour la resolution d'un probleme de Stokes.
%-----%
%Systeme de Stokes avec un ecoulement de Poiseuille solution (vi-
%  tesse nulle sur y = 0 et y = 1).
%Commande :
%
%      cd C :/EF_text/exemples/modeles, instruct1;
%

clear;
close all;

%Repertoires du code a ajouter au PATH :
cd('C :/EF');
path_EF;

%Repertoire de la fonction qui initialise la variable DONNEES :
cd('C :/EF_text/exemples/modeles/don');

%Handle sur la fonction d'initilisation de la variable DONNEES :
fdon = @don1;

%Initialisation de la variable DONNEES :
donnees = feval(fdon);

%Pre-calculs precedant le calcul de l'assemblage, initialise le
%  fichier de nom donne par donnees.nom_prep :
EF_Prep(donnees);

%Initialisation de l'itere de depart Uc a 0. Quand elle comporte
%  3 variables d'appels, la fonction INIT_TRANSF_CELL utilise le
%  fichier calcule par EF_PREPA de nom indique par la variable
```

5.15. MODELE DE FICHIER SCRIPT EXECUTANT LE PROGRAMME.

```
%  DONNEES.nom_prep :
Uc = init_transf_cell('init', [], donnees);

%Assemblage du systeme, conditions aux bord, prise en compte des
%  fonctions de moyenne nulle et elimination d'inconnues :
[S,f] = EF_assemblage(donnees,Uc);

%Resolution du systeme :
U = S\f;

%Mise de la solution U sous forme de vecteur cellule Uc :
U = full(U);%facultatif
Uc = init_transf_cell('ta2c',U,donnees);

%Exemple de sauvegarde. On sauvegarde la solution et maillage :
load(donnees.nom_mail,'X','Y','T');
save('C :/EF_text/exemples/modeles/Res/sol_don1', ...
      'donnees','Uc','T','X','Y');

%Exemple de traces. Trace de la premiere composante. La commande
%  LOAD est facultative ici car les variables deja chargees :
load('C :/EF_text/exemples/modeles/Res/sol_don1', ...
      'donnees','Uc','T','X','Y');

%Trace de u1 :
trac_solEd(donnees, Uc, 0, 1, 4);

%Sauvegarde du graphe dans une figure .PNG :
view(-10,40);
print('-dpng',['C :/EF_text/exemples/modeles', ...
              '/figures/instruct1.png']);

%Verification par comparaison avec une solution deja calculee :
Ucc = Uc;
```

```
load('C :/EF_text/exemples/modeles/comp/sol_don1', ...  
      'donnees','Uc','T','X','Y');  
  
for i=1 :3  
    disp(norm(Uc{i} - Ucc{i}));  
end;
```

5.16. MODELE DE FONCTION D'INITIALISATION DES DONNEES.

5.16. Modèle de fonction d'initialisation des données

Voici un exemple de fonction d'initialisation des données. Cette fonction est mise en œuvre par la commande

```
cd C :/EF_text/exemples/modeles, instruct1 ;
```

où `instruct1` est la fonction du Paragraphe 5.15 précédent.

Le fichier de fonction de données est donné à la page suivante. La variable d'entrée `i` de la fonction est ici facultative, elle est mise à titre d'exemple.

Fichier EF_text/exemples/modeles/don/don1.m

```
function donnees = don1(i) ;
%DON1. Initialisation de la variable DONNEES de description
%   des donnees d'un probleme EF.
%   La variable I est facultative ici. La variable DONNEES est
%   initialisee a la fin de la fonction DON1.
%-----%

%Chemin du repertoire donnees contenant la fonction DON1 ci-dessus,
%   utilisation uniquement dans DON1 :
nom_rep = 'C :/EF_text/exemples/modeles' ;

%----- NOM DES FICHIERS .MAT -----%
%Nom du fichier de calcul de donnees relatif aux espaces EF et aux
%   formules de quadrature :
nom_prep = [nom_rep, '/matri/mat_don1'] ;
%Nom du maillage, cree par la commande
%   cd C :/EF_text/exemples/maillages, creation_maillages
nom_mail = [nom_rep, '/mes/maillage1'] ;

%----- HANDLES SUR LES FICHIERS DE FONCTIONS UTILISATEURS -----%
%Stockage dans une variable du nom du repertoire courant, avant
%   l'initialisation de variables a des HANDLES de fonctions
%   appartenant a d'autres repertoires :
rep = pwd ;
%Repertoire contenant les fonctions :
cd([nom_rep, '/equ']) ;

%CONDITIONS AUX BORD.
%Fichier des conditions aux limites :
fcb = @cb_don ;

%EQUATIONS PREMIER MEMBRE.
```


5.16. MODELE DE FONCTION D'INITIALISATION DES DONNEES.

```

%Sur le domaine :
feq_Dom = @eq_Dom ;
%Sur la frontiere :
feq_Fr = @eq_Fr ;
%Sur les aretes du maillage :
feq_Ad = @eq_Ad ;

%EQUATIONS SECOND MEMBRE.
%Sur le domaine :
fscm_Dom = @scm_Dom ;
%Sur la frontiere :
fscm_Fr = @scm_Fr ;
%Sur les aretes du maillage :
fscm_Ad = @scm_Ad ;

%Methode de Lesaint-Raviart :
fLR1 = @eq_LR1 ;
%Autres fonctions pour Lesaint-Raviart :
fLR2 = @eq_LR2 ;
%Retour au repertoire courant :
cd(rep)

%-----%
% INITIALISATION DE LA VARIABLE D : POUR LE CHAMP D DE DONNEES.D %
%-----%

%DEFINITION DE l'ESPACE EF ET CHOIX DES FORMULES DE QUADRATURE :
%-----%
%Description de l'espace EF produit.
%Choix des types d'espaces EF FI composant l'espace EF produit
% avec ici :
% F1 = P1, F2 = P2, espaces EF respectivement P1, P2 continus :
D.E = [1,2] ;
%Derivation, 0, x, y (choix invariable dans le code EF) :
```

```

D.der = [0,1,2] ;
%Choix de l'espace EF produit, ici E = F2 x F2 x F1 :
D.f = [2,2,1] ;
%Formule d'integration sur les triangles, choisie d'ordre 8 ici :
D.intT = 8 ;
%Formule d'integration sur les aretes, choisie d'ordre 6 ici :
D.intA = 6 ;

%INCONNUES PONCTUELLES A ENLEVER (CHAMP FACULTATIF).
%-----%
%Dans le probleme etudie, la pression est definies a une constante
%  pres. Pour en tenir compte, on enleve une composante de la
%  pression. Ici c'est la derniere composante de la pression qui
%  est choisie (pour ce choix, voir la doc.).
%  a) Numero d'inconnue sur laquelle porte l'elimination :
D.inc_point{1,1} = 3 ;
%  b) Numeros de DDL sur lesquels portent l'elimination, vecteur
%  ligne ligne ou valeur 'end' qui indique le choix de la
%  derniere composante :
D.inc_point{1,2} = 'end' ;
%  c) Valeur de la composante de la solution au DDL elimine, pour
%  reconstituer ensuite la solution dans la base de l'espace EF :
D.inc_point{1,3} = 0 ;

%AUTRES INCONNUES PONCTUELLES A ENLEVER (EXEMPLE).
%Autre exemple d'ecriture du cas ci-dessus :
%D.inc_point(2, :) = {3,'end',0)

%INCONNUES A ELIMINER (CHAMP FACULTATIF).
%-----%
%  Pour des inconnues satisfaisant les memes equations et suppo-
%  sees identiques :
%Champ facultatif ou pose a [] :
D.inc_cell = [] ;

```

5.16. MODELE DE FONCTION D'INITIALISATION DES DONNEES.

```
%AUTRES INCONNUES A ELIMINER (EXEMPLE).
%On utilise un vecteur colonne qui indique deux des composantes de
% l'inconnue concernees :
%D.inc_cell = [1;2] ;

%AUTRES INCONNUES A ELIMINER (EXEMPLE).
% Ici l'exemple concerne une inconnue avec plus de 4 composantes.
% Ici, on suppose que D.inc_cell = [1;2], ce qui donne, avec la
% commande ci-dessous, comme inconnues identiques les inconnues
% numero 1,2 et 4 :
%D.inc_cell( :,2) = [1;4] ;

%METHODE DE LESAIN-T-RAVIART (CHAMP FACULTATIF).
%-----%
%Initialisation a des HANDLES de fonctions representant les champs
% de vecteurs suivant lesquels on derive et numeros de composantes
% des inconnues principales concernees, mis sous forme de vecteur
% ligne :
%Premiere derivation :
D.LR(1, :) = {fLR1, [1,2]};
%Autres derivations :
D.LR(2, :) = {fLR2, [3] };
%-----%
%
%          FIN DE L'INITIALISATION DE LA VARIABLE D
%
%-----%

%-----%
%
%          INITIALISATION DE LA VARIABLE DONNEES
%
%-----%

%----- NOM DES FICHIERS .MAT -----%
%Noms des fichiers .mat de maillage et de donnees relatives aux
% espaces EF et aux formules d'integration :
```

```

donnees.nom_mail = nom_mail ;
donnees.nom_prep = nom_prep ;

%----- CHAMPS FACULTATIFS -----%
%Fonctions utilisees par EF_assemblage :
donnees.feq_Dom = feq_Dom ;
donnees.feq_Fr = feq_Fr ;
donnees.feq_Ad = feq_Ad ;
donnees.fscm_Dom = fscm_Dom ;
donnees.fscm_Fr = fscm_Fr ;
donnees.fscm_Ad = fscm_Ad ;
donnees.fcb = fcb ;

%----- VARIABLE DECRIVANT LES ESPACES EF -----%
%Description des espaces d'EF :
donnees.D = D ;

%-----CHAMPS DE DONNEES CREES PAR L'UTILISATEUR-----%
%Pas de champs (utilisateurs) supplementaires ici.

%-----%
%          FIN DE L'INITIALISATION DE LA VARIABLE DONNEES          %
%-----%

%----- DIVERS -----%
%
%---- DOMAINE ET FRONTIERES NUMEROTEES EN SENS TRIGO. INVERSE ----%
% Le domaine est le carre unite [0,1] x [0,1], sa frontiere se de-
% compose en 4 morceaux de frontiere decrits par la variable
% FRONT du fichier .MAT de maillage :
%
%          ___Front{1}___
%          |               |
%          |               |

```

5.16. MODELE DE FONCTION D'INITIALISATION DES DONNEES.

```
%   Front{4} |               | Front{2}
%           |               |
%           |-----|
%           Front{3}
%
%----- FIN DIVERS -----%
```

5. APPENDICE

5.17. Création de maillage

Pour la création de maillages *cf.* aussi le Paragraphe 2.4.

1) Utilisation de `pdetool`.

- Ouvrir l'utilitaire de maillage avec la commande :

```
pdetool ;
```

Cela ouvre une fenêtre intitulée `PDE Toolbox`.

- Ensuite, ébaucher «à la main» avec le pointeur de la souris un maillage. On pourra utiliser le menu déroulant avec :

```
menu/Options/Axes Limits
```

ainsi que

```
menu/Options/Axes Equal
```

pour modifier les axes. On obtient alors un maillage avec des cotes approximatives à réajuster, ce que nous verrons plus loin.

- A partir du menu, sauvegarder le résultat dans un fichier `.m` avec :

```
menu/File/Save As
```

par exemple, pour la suite de notre propos, dans le fichier

```
C :/EF_text/exemples/modele_a/mes/maillage_init.m.
```

- Quitter l'utilitaire de maillage.
- Ouvrir le fichier sauvegardé. Sur l'exemple, c'est donc le fichier :

```
C :/EF_text/exemples/modele_a/mes/maillage_init.m.
```

Le fichier commence alors par

```
function pdemodel
```

- Remplacer cet en-tête en ajoutant les variables de sortie `pde_fig` et `ax` (cette dernière variable est facultative) et en modifiant, au choix, le nom de la fonction. Par exemple :

```
function pde_fig = maillage ;
```

ou bien

```
function [pde_fig,ax] = maillage ;
```

En sortie `pde_fig` et `ax` sont des valeurs de pointeurs d'objet calculé dans la fonction `maillage`. La variable `pde_fig` pointe sur un objet de type `figure` et la variable `ax` pointe sur un objet de type `axes` (on obtient les propriétés de ces objets par la fonction `get`).

- Réajuster le maillage en mettant les cotes souhaitées dans le *script* de la fonction : cela se fait en modifiant les arguments des fonctions `pdirect`, `pdepoly`, *etc.* suivant le maillage représenté.

5.17. CREATION DE MAILLAGE.

- Sauver le fichier rectifié, en utilisant le même nom ou un autre nom. Pour notre exemple, le fichier est sauvegardé sous le nom :

C :/EF_text/exemples/modele_a/mes/maillage.m.

- Fermer la fenêtre PDE Toolbox et vérifier le résultat en reconstituant la figure maillé dans la fenêtre PDE Toolbox. Pour l'exemple, cela s'effectue par la commande

```
cd C :/EF_text/exemples/modele_a/mes, maillage
```

ce qui donne le maillage suivant :

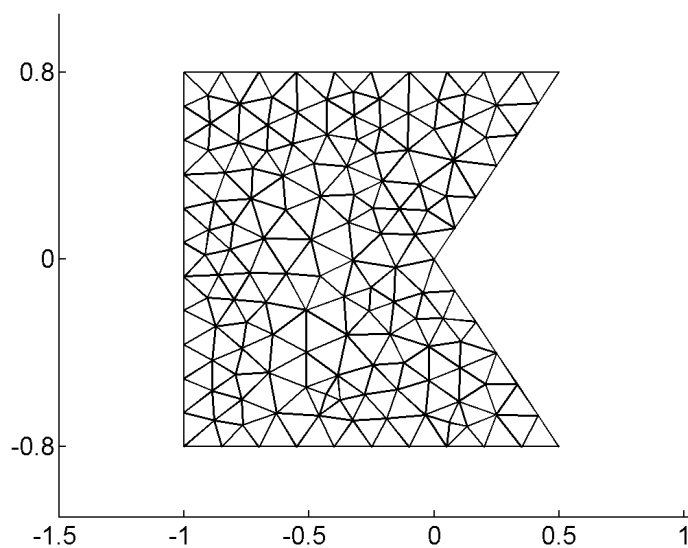


Figure A-2. Maillage donné par la fonction `maillage` de l'exemple.

2) Passer de la structure du maillage obtenue par `pde-tool` à celle de la structure de données du code

Utilisation de `bmailP1`.

Cette fonction permet de passer d'une structure de maillage à l'autre ainsi que de regrouper des morceaux de frontières entre elles. Pour cela la fonction `bmailP1` nécessite de respecter l'ordre et l'orientation des frontières. On pourra aussi utiliser la fonction `bmailP1_b` décrite à la suite qui elle ne tient pas compte de cette orientation et est plus simple d'emploi.

L'ordre choisi pour le regroupement de morceaux de frontières doit aussi coïncider avec l'orientation de ces frontières ce qui n'est pas toujours possible, notamment lorsque deux frontières contigües que l'on veut regrouper ont des orientations opposées. D'autre part, lorsque des morceaux de frontière du maillage ne comportent pas de sommets du maillage en dehors de leur extrémités, la fonction `bmailP1` peut ne pas prendre en compte ce maillage. Dans ces deux cas on utilisera alors la fonction `bmailP1_b`.

5. APPENDICE

- Traitement des frontières : il faut repérer les numéros de frontières et leur orientation. Pour cela, sur la fenêtre `PDE toolbox` comportant la figure on utilise le menu :

- Menu/Boundary/Boundary mode

puis

- Menu/Boundary/Show Edge Labels,

ce qui donne sur l'exemple

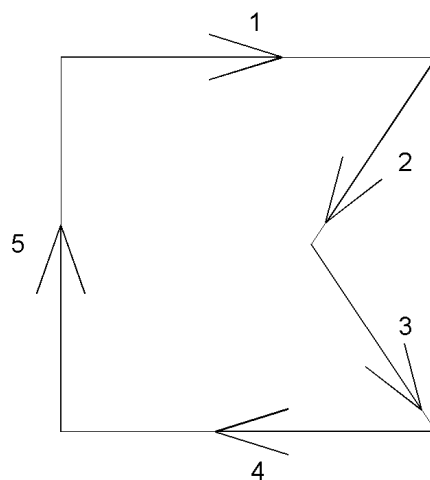


Figure A-3. Maillage donné par la fonction `maillage`.

- Regarder les numéros de frontières et leur orientation (flèches) : on peut regrouper des morceaux de frontières contigus et de même orientation.

- Ecrire une fonction analogue à la fonction `mail_maillage` ci-dessous pour traiter l'exemple.

```
function [e,fr,dom] = mail_maillage ;
```

```
%Nom du fichier final obtenu a partir de pdetool
```

```
e = 'maillage' ;
```

```
%Traitement des frontieres : regroupement de frontieres
```

```
fr = [{[1]}, {[2,3]}, {[4,5]}] ;
```

```
%Domaine de raffinement (ici pas de raffinement)
```


5.17. CREATION DE MAILLAGE.

`dom = [] ;`

qui a pour chemin :

`C :/EF_text/exemples/modele_a/mes/mail_maillage.m.`

Dans cet exemple on pourra utiliser d'autres regroupements de frontières comme :

`fr = [{[1,2,3]},{[4,5]}] ;`

mais pas comme

`fr = [{[3,2,1]},{[4,5]}] ;`

car il faut suivre le sens des flèches, ni comme

`fr = [{[3,1]},{2},{4,5}] ;`

car les frontières doivent être contigües.

Avec cette fonction, la frontière 1 du maillage final est ainsi composée de la frontière 1 de la figure A-3, de la frontière 2 qui est la réunion des frontières 2 et 3 du maillage A-3 et de la frontière 3, réunion des frontières 4 et 5. Le résultat est donné à la figure A-4 ci-dessous.

- Création des variables du maillage pour le code EF. Sur l'exemple les variables sont créées par la commande :

`[T,X,Y,Noeu,Front] = bmailP1('mail_maillage')`

- Fin : création d'un fichier *maillage.mat* contenant le maillage dans le répertoire courant

`save('maillage','T','X','Y','Noeu','Front') ;`

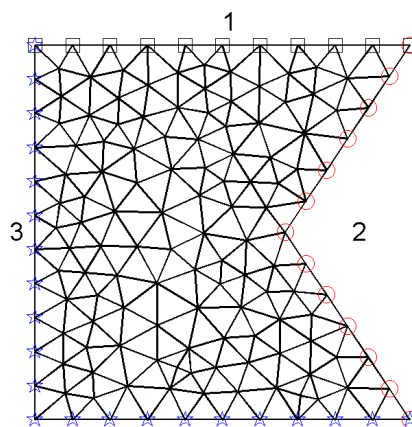


Figure A-4. Maillage réalisé par la fonction `maillage` et numéros de frontière.

5. APPENDICE

Utilisation de `bmailP1_b`.

On utilisera `bmailP1_b` si des frontières consécutives n'ont pas la même flèche. Cette fonction donne, comme la fonction `bmailP1`, une variable `Front` dont chaque cellule numéro i décrit les sommets successifs de la frontière obtenue par réunion des frontières indiquées par `fr{i}`. Cependant le résultat sur `Front` peut cependant changer par rapport à `bmailP1`. Un exemple d'utilisation de `bmailP1_b` est donné avec la fonction suivante :

```
function [e,fr,dom] = mail_maillage_b ;  
%Nom du fichier final obtenu a partir de pdetool  
e = 'maillage' ;
```

```
%Traitement des frontieres  
fr = [{[1]},{[3,2]},{[4,5]}] ;
```

```
%Domaine de raffinement
```

```
dom = [] ;
```

```
de chemin :
```

```
C :/EF_text/exemples/modele_a/mes/mail_maillage_b.m.
```

et la commande

```
[T,X,Y,Noeu,Front] = bmailP1_b('mail_maillage_b') ;
```

qui donnent le même résultat pour `Front` que `bmailP1` dans le cas précédent, à l'exception d'un ordre inverse des sommets pour `Front{2}`.

3) Raffinement.

La fonction suivante donne un exemple d'utilisation de la variable `Dom` pour le raffinement de maillage. Les zones de raffinement successifs sont indiquées par des polygones qui sont ici deux rectangles.

```
function [e,fr,dom] = mail_maillage_raf ;
```

```
%Nom du fichier final obtenu a partir de pdetool
```

```
e = 'maillage' ;
```

```
%Traitement des frontieres
```

```
fr = [{[1]},{[2,3]},{[4,5]}] ;
```

```
%Domaines de raffinements successifs :
```

```
%Zone du premier raffinement : rectangle de sommets (x(i),y(i)) :
```

5.17. CREATION DE MAILLAGE.

```
x = [0,0.5,0.5,0] ;  
y = [-0.8,-0.8,0.8,0.8] ;  
dom{1,1} = x ;  
dom{1,2} = y ;  
  
%Zone du second raffinement : rectangle de sommets (x(i),y(i)) :  
x = [0.3,0.5,0.5,0.3] ;  
y = [-0.8,-0.8,0.8,0.8] ;  
dom{2,1} = x ;  
dom{2,2} = y ;  
La commande  
    [T,X,Y,Noeu,Front] = bmailP1_b('mail_maillage_raf') ;  
donne alors le maillage de la figure A-5 suivante.
```

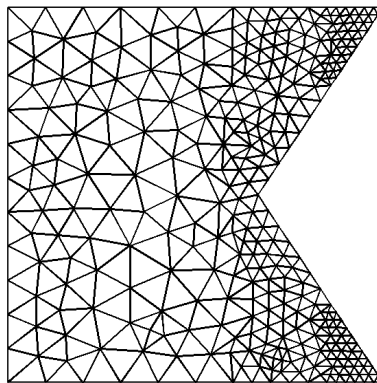


Figure A-5. Maillage raffiné donné par la fonction `maillage_raf`.

4) La variable `pde_fig`.

Cette variable permet de récupérer la valeur du pointeur de l'objet de type figure qui contient la partie de la structure du maillage et de la géométrie du domaine nécessaire au code.

Pour récupérer cette structure on utilise les commandes suivantes :

```
pde_fig = maillage ;  
[p,e,t,g]= export_mail(pde_fig) ;
```

5. APPENDICE

Ces quatre variables **p**, **e**, **t**, **g** s'obtiennent aussi «manuellement» de la manière suivante. On utilise la commande

maillage ;

avec le fichier **maillage** nouvellement créé. Ensuite, dans la fenêtre **PDE Toolbox** où figure le nouveau maillage, on sélectionne le menu

- **Mesh/Export Mesh**

qui donne les valeurs de **p**, **e**, **t** pour les données du maillage (coordonnées des points, numéros de sommets des frontières et numéros des sommets du maillage triangles par triangles) et par

- **Menu/Boundary/Export Decomposed geometry, Boundary Conds**

pour la valeur de la variable **g** de géométrie du domaine. Dans le code la variable **g** est utilisée par les fonctions **bmailP1** et **bmailP1_b** pour le raffinement.

Table des matières

Introduction.

1. Description des types de problèmes assemblés.

- 1.1. Une formulation abstraite du problème.
- 1.2. Type des formes bilinéaires et linéaires prises en compte dans le code.
- 1.3. Exemples de matrices de définition de l'EDP.
- 1.4. Espaces d'EF utilisés dans le code, degrés de liberté (*ddl*), bases.
- 1.5. Bases et représentation des fonctions de l'espace EF produit E .

2. Mode d'emploi pour l'utilisation du programme.

- 2.1. Un exemple sur le problème du Laplacien. Problème ($\mathcal{P}_1$).
 - 2.1.2. Valeur des applications a_D et b_D du Paragraphe 1.2.
 - 2.1.2.1. Valeur des tableaux de cellules définissant le problème.
 - 2.1.2.2. Autres écritures des tableaux de cellules pour la forme bilinéaire : forme non factorisée.
 - 2.1.2.3. Autres écritures des tableaux de cellules pour la forme bilinéaire : forme factorisée.
 - 2.1.2.4. Autres écritures des tableaux de cellules pour la forme linéaire.
 - 2.1.3. Un fichier script illustrant l'utilisation du code.
 - 2.1.4. Résolution du problème et description de la structure de données.
 - 2.1.4.1. La fonction d'initialisation des données.
 - 2.1.4.2. Fichiers de fonctions pour la forme bilinéaire.
 - Fichiers comportant plusieurs pages.
 - 2.1.4.3. Fichiers de fonctions pour le second membre (formes linéaires).
 - Fichiers comportant plusieurs pages.
 - 2.1.4.4. Fichiers de fonctions pour les conditions au bord.
 - 2.1.4.5. Valeurs prises par la variable `donnees`.
 - 2.1.4.6. Un exemple de compilation C.
 - 2.1.5. Résolution du Laplacien sur d'autres exemples.
 - 2.1.6. Résolution du Laplacien avec condition de Neumann. Espaces quotients.
 - 2.1.7. Résolution du Laplacien avec un Laplacien au second membre.

- 2.2.** Description de la structure de données utilisée par le code.
 - 2.2.1.** Description des champs de la variable de type `struct` décrivant les données.
 - 2.2.2.** Description de la structure du maillage.
 - 2.2.3.** Description du fichier décrivant les espaces d'EF et l'intégration.
 - 2.2.4.** Syntaxe des fonctions décrivant l'EDP (handles du Tableau 2.2.1-1b.)
 - 2.2.4.1.** Exemple de fonction définissant a_Ω (champ `feq_Dom`).
 - 2.2.4.2.** Exemple de f^n définissant le s^{cd} membre b_Ω (champ `fscm_Dom`).
 - 2.2.4.3.** Exemple de f^n pour les conditions aux limites (champ `fc_b`).
 - 2.2.5.** Variables de sorties et variables calculées dans `EF_Assemblage`.
- 2.3.** Description des différents espaces d'EF programmés dans le code.
 - 2.3.1.** Les types d'EF et description algorithmique.
 - 2.3.2.** Description de divers espaces d'EF sur un maillage exemple.
 - 2.3.3.** Ordre local des *ddl* et construction des fonctions de base.
- 2.4.** Maillages et utilisation de `pdetool`.
- 2.5.** Traitement des solutions et vérification des données.
- 2.6.** Exemples de système d'EDP.
 - 2.6.1.** Résolution approchée du problème $(\mathcal{P}_2)$.
 - 2.6.1.1.** Formulation variationnelle du problème.
 - 2.6.1.2.** Le problème approché $(\mathcal{P}_{2h})$.
 - 2.6.1.3.** Résolution du problème approché.
 - 2.6.1.4.** Récapitulatif des fonctions utilisées pour la résolution de $(\mathcal{P}_2)$.
 - 2.6.1.5.** Autres approximations EF de $(\mathcal{P}_2)$, forme factorisée.
 - 2.6.2.** Deuxième exemple : le système avec conditions de Neumann $(\mathcal{P}'_2)$.
 - 2.6.3.** Troisième exemple : autres systèmes, avec conditions de Dirichlet données sous forme de combinaisons linéaires, problèmes $(\mathcal{P}''_2)$ et $(\mathcal{P}'''_2)$.
- 2.7.** Exemples d'équations avec un terme de transport.
 - 2.7.1.** Résolution approchée du problème $(\mathcal{P}_3)$.
 - 2.7.1.1.** Résolution de $(\mathcal{P}_3)$ sans décentrage.
 - 2.7.1.2.** Résolution de $(\mathcal{P}_3)$ avec ajout d'un décentrage.
 - 2.7.1.3.** Résolution de $(\mathcal{P}_3)$ avec la méthode de Lesaint-Raviart.
 - 2.7.2.** Résolution approchée du problème $(\mathcal{P}'_3)$.
 - 2.7.3.** Résolution du système $(\mathcal{P}''_3)$ avec la méthode de Lesaint-Raviart.

- 2.7.4. Récapitulatif des fonctions utilisées pour le premier problème, le Problème $(\mathcal{P}_3)$.
- 2.8. Exemples sur l'équation d'ondes. Problème $(\mathcal{P}_4)$.
- 2.9. Exemples sur le système de l'élasticité linéaire.
 - 2.9.1. Formulation variationnelle du système.
 - 2.9.2. Valeur des matrices du système.
 - 2.9.3. Le problème approché.
 - 2.9.4. Résolution du problème $(\mathcal{P}_{5h}^{(1)})$.
 - 2.9.5. Résolution du problème $(\mathcal{P}_{5h}^{(2)})$.
 - 2.9.6. Résolution du problème $(\mathcal{P}_{5h}^{(3)})$.
 - 2.9.7. Résolution du problème $(\mathcal{P}_{5h}^{(4)})$.
 - 2.9.8. Résolution de $(\mathcal{P}_{5h}^{(1)})$ avec d'autres conditions au bord.
- 2.10. Un exemple sur le système de Stokes.
 - 2.10.1. Formulation variationnelle du problème de Stokes.
 - 2.10.2. Valeur des matrices du système.
 - 2.10.3. Le problème de Stokes approché.
 - 2.10.4. Résolution de $(\mathcal{P}_6^{(1)})$ (Poiseuille).
 - 2.10.5. Résolution de $(\mathcal{P}_6^{(2)})$ (Poiseuille, pression donnée) et $(\mathcal{P}_6^{(2')})$.
 - 2.10.6. Résolution de $(\mathcal{P}_6^{(3)})$ et $(\mathcal{P}_6^{(3')})$ (condition de Neumann sur l'axe de symétrie).
 - 2.10.7. Résolution de $(\mathcal{P}_6^{(4)})$ $(\mathcal{P}_6^{(4)})$ et $(\mathcal{P}_6^{(4')})$. Autres types de conditions aux limites. Conditions sur les composantes normales et tangentielles.
 - 2.10.8. Problème $(\mathcal{P}_6^{(5)})$. Cavité entraînée.
 - 2.10.9. Problème $(\mathcal{P}_6^{(6)})$. Calcul des lignes de courant.
 - 2.10.10. Problème $(\mathcal{P}_6^{(7)})$. Problème de Stokes non linéaire.
 - 2.10.10.1. Résolution par une méthode de point fixe.
 - 2.10.10.2. Résolution par la méthode de Newton.
 - 2.10.10.3. Valeur des matrices du système pour la méthode de Newton.
 - 2.10.10.4. Calcul des coefficients pour la méthode de Newton.
- 2.11. Utilisation d'EF adaptatifs.
 - 2.11.1. Calcul de normes.

TABLE DES MATIERES

- 2.11.2.** Fonction de raffinement de maillage.
- 2.11.4.** Un exemple sur un système.
- 2.11.3.** Un exemple sur le Laplacien.
 - 2.11.3.1.** Description du fichier d'instructions *instruct1.m*.
 - 2.11.3.2.** Résolution adaptative du Laplacien : résultats.
- 2.11.5.** Un exemple de Laplacien posé sur un domaine en forme de L.
- 2.11.7.** Cas d'un problème non linéaire : loi puissance pour un fluide.
 - 2.11.7.1.** Résolution par point fixe.
 - 2.11.7.2.** Résolution par la méthode de Newton.
- 2.11.8.** Etude d'un écoulement de type viscoélastique.
 - 2.11.8.1.** Définition du problème. Le problème $(\mathcal{P})$.
 - 2.11.8.2.** Approximation EF continue de $(\mathcal{P})$ par $(\mathcal{Q}_h)$.
 - 2.11.8.3.** Approximation EF discontinue des tenseurs.
 - 2.11.8.4.** Méthode de point fixe couplé pour la résolution de $(\mathcal{Q}_h)$.
 - 2.11.8.5.** Les estimateurs locaux utilisés pour le problème.
 - 2.11.8.6.** Deux exemples d'écoulements.
 - 2.11.8.7.** Résultats numériques.
 - 2.11.8.8.** Les fichiers de fonctions.
 - 2.11.8.9.** Calcul des matrices pour le problème d'Oldroyd.
 - 2.11.8.10.** Matrices et opérateurs attachés au problème d'Oldroyd.
- 2.12.** Eléments finis courbes.
 - 2.12.1.** Les fonctions utilisées pour les EF courbes.
 - 2.12.1.1.** Les fonctions d'assemblage.
 - 2.12.1.2.** Les fonctions de tracé de calcul des *ddl*, d'interpolation aux *ddl*.
 - 2.12.1.3.** Calcul de norme.
 - 2.12.2.** Description des fonctions sur l'exemple d'une EDP sur un triangle.
 - 2.12.3.** Exemple d'un Laplacien sur un domaine avec un coin rentrant.
 - 2.12.4.** Ecoulements de Couette d'un écoulement Newtonien.

3. Quelques outils mathématiques

3.1. Assemblage.

3.1.1. Représentation des fonctions à valeur matricielles.

3.1.1.1. Fonctions du code en relation avec le Paragraphe 3.1.1.

3.1.2. Assemblage. Calcul des matrices élémentaires : cas scalaire.

3.1.2.1. Cas des formes bilinéaires.

3.1.2.2. Cas des formes linéaires.

3.1.2.3. Cas des arêtes.

3.1.2.4. Fonctions du code en relation avec le Paragraphe 3.1.2.

3.1.3. Assemblage. Calcul des matrices élémentaire. Cas des systèmes.

3.1.3.1. Assemblage global.

3.1.3.2. Cas des formes linéaires : matrices élémentaires et assemblage global.

3.1.3.3. Fonctions du code en relation avec le Paragraphe 3.1.3.

3.2. Traitement des conditions au bord.

3.2.1. Conditions au bord.

3.2.2. Détermination des normales et tangentes dans le cas des points anguleux.

3.2.3. Fonctions du code en relation avec le Paragraphe 3.2.

3.3. Exemples de formules de quadrature.

3.3.1. Intégration sur un triangle.

3.3.2. Intégration sur les arêtes d'un triangle.

3.3.3. Fonctions du code en relation avec le Paragraphe 3.3.

3.4. Exemple d'expression des fonctions de base et de leurs dérivées.

3.5. Assemblage pour la méthode de Lesaint-Raviart.

3.6. Espaces d'éléments finis courbes.

3.6.1. Interpolation.

3.6.2. Formules de quadrature sur les triangles.

3.6.3. Formules de quadrature sur les arêtes.

3.6.4. Formules de dérivations.

3.6.5. Choix de la normale et de la tangente dans le cas d'un point anguleux.

3.7. Bibliographie.

3.7.1. Bibliographie éléments finis.

- 3.7.2.** Références bibliographiques relatives au problème d'Oldroyd et aux calculs effectués par un code similaire à celui présenté ici.

4. Description du code

- 4.1.** Fonctions et algorithme général du code.
 - 4.1.1.** Liste des répertoires et fonctions utilisés par le code.
 - 4.1.2.** Algorithme général du code.
- 4.2.** Principales variables utilisées par le code.
- 4.3.** Description des fonctions et algorithmes.

5. Appendice

- 5.1.** Ajoût des répertoires du code aux chemins de recherche.
- 5.2.** Variables du code (récapitulatif de tableaux).
- 5.3.** Paramètres des fonctions décrivant les EDP et le second membre, récapitulatif de tableaux).
- 5.4.** Paramètres des fonctions décrivant les conditions au bord (récapitulatif de tableaux).
- 5.5.** Modèle de fonction pour les conditions au bord.
- 5.6.** Modèle de fonction décrivant les EDP sur le domaine : forme factorisée.
- 5.7.** Modèle de fonction décrivant le second membre sur le domaine : forme factorisée.
- 5.8.** Modèle de fonction décrivant les EDP sur la frontière : forme factorisée.
- 5.9.** Modèle de fonction décrivant le second membre sur la frontière : forme factorisée.
- 5.10.** Modèle de fonction décrivant les EDP sur les arêtes du maillage : forme factorisée.
- 5.11.** Modèle de fonction décrivant le second membre sur les arêtes du maillage : forme factorisée.
- 5.12.** Modèle de fonction pour la méthode de Lesaint-Raviart.
- 5.13.** Modèle de fonction décrivant les EDP sur le domaine : forme non factorisée.
- 5.14.** Modèle de fonction décrivant le second membre sur le domaine : forme non factorisée.
- 5.15.** Modèle de fonction exécutant le programme.
- 5.16.** Modèle de fonction d'initialisation des données.
- 5.17.** Création de maillage.

6. Résumé.

RESUME

Résumé. Nous présentons dans ce document un code qui met en œuvre la méthode des éléments finis (MEF) pour la résolution approchée des équations aux dérivées partielles (EDP) en dimension 2 (2-D). Le mode d'utilisation de ce code est illustré par des exemples issus de divers types de problèmes d'EDP. Ce code permet notamment la mise en œuvre de méthodes d'EF pour l'approximation d'écoulements de fluides viscoélastiques.

L'écriture de ce code est issue d'un code EF initialement construit pour la mise en œuvre de méthodes numériques développées pour l'approximation d'écoulements de fluides viscoélastiques, notamment pour la prise en compte du modèle de Maxwell linéarisé et pour le calcul adaptatif sur le modèle d'Oldroyd. Nous avons repris l'approche algorithmique du code initial, celle-ci facilitant la programmation de formulations variationnelles relativement complexes, surtout au niveau de la partie «fonctions test», et l'avons placée dans un cadre plus large. Le code actuel permet alors de traiter des formulations variationnelles et des espaces d'EF plus généraux que dans son écriture initiale.

L'utilisation du code est basé sur l'emploi d'une fonction nommée `EF_assemblage` qui effectue l'assemblage du problème EF à partir notamment de la donnée d'une description du problème EF par des formes bilinéaires et linéaires. Cette fonction a comme variable d'entrée une variable de type `struct` dont les champs décrivent la structure de données du problème avec principalement : le nom du fichier de maillage, le nom des fonctions définissant les conditions au bord, le nom des fichiers décrivant les formes bilinéaires et les formes linéaires intervenant dans la formulation EF du problème, le type des EF choisis, continus ou discontinus usuels, auxquels peuvent se rajouter des fonctions de type «bulles», et enfin le choix des formules d'intégration sur les triangles et sur les arêtes de triangles.

Nous avons détaillé, émaillé d'exemples, la structure de maillage utilisée, la classe d'espaces d'EF employés (espaces d'EF polynômiaux, continus ou discontinus, complétés le cas échéant par des fonctions de type bulles permettant de satisfaire des conditions de convergence de la méthode des EF) ainsi que la structure des *ddl* (degrés de liberté) mise sur ces espaces.

Des exemples d'utilisation de la méthode des EF et du code EF sont donnés pour la résolution du problème du Laplacien, du problème du transport, de l'équation d'ondes, de l'élasticité, de problèmes d'écoulements de fluides newtoniens (Stokes) ou quasi-newtoniens (loi puissance) avec, dans ce cas, une résolution par point fixe ou avec la méthode de Newton, ainsi que la résolution de problèmes d'écoulements de fluides viscoélastiques (fluides d'Oldroyd, de Maxwell).

Parmi ces exemples, la méthode des EF est aussi mise en œuvre dans le cadre des EF adaptatifs où le maillage subit des raffinements successifs pour satisfaire des critères de convergence de la solution définis par l'utilisateur. Pour cela l'utilisateur construit une fonction qui décrit l'estimateur d'erreur *a posteriori* choisi, le calcul d'erreur étant effectué dans cette fonction à partir de la solution approchée obtenue sur le maillage à raffiner.

RESUME

Nous abordons aussi, illustré par des exemples, le cadre des EF dits «courbes» pour la prise en compte de domaine à frontière courbe. Pour ce type d'EF, la fonction qui réalise l'assemblage est cette fois-ci la fonction **EF_assemblage_t**.

Nous décrivons les fonctions utilisées par le code EF (hors exportation de maillage, adaptatif et EF courbes) en détaillant les principaux algorithmes et nous examinons les notions mathématiques entrant en jeu dans la construction du code : techniques utilisées pour l'assemblage des équations scalaires, des systèmes, intégration approchée, traitement d'espaces quotients, traitement des conditions au bord, construction des estimateurs d'erreur locaux *a priori*, transformation pour le passage aux espaces d'EF courbes.

Institut Camille Jordan
Domaine Scientifique de la Doua
Bâtiment du Doyen Jean Braconnier
43 Bd du 11 novembre 1918
69622 Villeurbanne
<http://math.univ-lyon1.fr>

