Analyse matricielle

Pierre Gosselet

Master MS2SC
Analyse matricielle

version dangereusement non-définitive
merci de signaler erreurs et lacunes

Pierre Gosselet
LMT – CNRS - École Normale Paris-Saclay
pierre.gosselet@ens-paris-saclay.fr

Septembre 2018

Table des matières

Chapitre 1

Calcul numérique

1.1 Nombres à virgule flottante

1.1.1 Les nombres à virgule flottante

Un nombre à virgule flottante est de la forme $x = s.m.\beta^e$: s est le signe de x, m est la mantisse, β la base (en pratique 2) et e l'exposant. La mantisse m est décrite par au plus p chiffres en base β (m est un nombre à virgule) et $e_{\min} \leqslant e \leqslant e_{\max}$, e est lui aussi stocké en base β sous la forme de d chiffres et un signe. Un dernier détail consiste à rendre unique la représentation des nombres (parce que $3.140\ 10^0 = 0.314\ 10^1$). Une solution, en base deux, est de toujours imposer que le premier chiffre de la mantisse soit 1 (d'ailleurs ce nombre n'est pas stocké, c'est le bit de tête implicite), on parle de nombre normalisé. Cela limite la précision inférieure des calculs. Une solution est de pouvoir passer en représentation dénormalisée (ce qui permet des mantisses « plus petites »).

La figure 1.1 donne une représentation de nombres flottants obtenus pour des flottants du type $x = 0.d_1d_2d_32^e$ où les d_i sont des chiffres en base 2 et e est stocké sur 2 bits ($-1 \leqslant e \leqslant 2$). On voit que les flottants ne sont pas équirépartis dans $\mathbb{R}$.

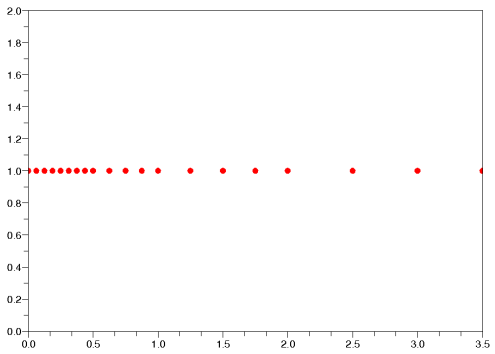


FIGURE 1.1 – Exemple de répartion de flottants

Soit $\mathbb{F} \subset \mathbb{Q}$ l'ensemble des flottants. Malheureusement, cet espace n'a pas de belle structure algébrique quand on le munit des opérations usuelles. En particulier, il n'est pas stable par les opérations usuelles. Par exemple 1 et $(10)_{decimal} = (1010)_{binaire}$ sont des flottants mais $(1/10)_{decimal} = 0.0001100110011\ldots$ n'est pas flottant. On doit donc systématiquement ajouter une étape d'arrondi pour pouvoir représenter le résultat d'une opération dans $\mathbb{F}$. Outre des problèmes de précison, cet arrondi entraîne la perte de certaines propriétés notamment les opérations ne sont plus associatives (ni distributives). Cela signifie que l'ordre dans lequel on conduit les opérations est primordial. Par exemple :

$$\text{en double : } 0.1 + (10^{20} - 10^{20}) = 0.1, \qquad (0.1 + 10^{20}) - 10^{20} = 0. \tag{1.1}$$

Pour les flottants, si on ne met pas de parenthèses pour forcer l'ordre d'évaluation des opérations, on ne peut pas garantir la reproductibilité d'un résultat.

1.1.2 La norme IEEE 754

Quatre formats en base 2 sont prévus (voir table 1.1) : simple précision (correspondant généralement au type *float* en C), simple précision étendue (obsolète), double précision (correspondant généralement au type *double* en C) et double précision étendue (*extended* en anglais). La quadruple précision a été introduite plus récemment. La norme prévoit également les nombres spéciaux $-\infty$, $+\infty$ et NaN (*not a number*) sur lesquels les opérations standards sont définies (+,-,*,/). Le zéro est signé ($1/(-0) = -\infty$).

format	taille	précision	$e_{\min}$	$e_{\max}$	valeur max
simple	32	23+1 bits	-126	+127	$3.403...10^{38}$
double	64	52+1 bits	-1022	+1023	$1.798...10^{308}$
extended	$\geqslant 79$	$\geqslant 63+1$ bits	$\leqslant -16382$	$\geqslant 16383$	$1.190...10^{4932}$
quadruple	128	$112+1$ bits	-16382	16383	$1.190...10^{4932}$

TABLE 1.1 – Norme IEEE 754

Des drapeaux (*flags*) sont prévus, auxquels il est possible d'associer des exceptions :
- drapeau inexact quand le calcul a nécessité un arrondi,
- drapeau opération invalide quand le calcul donne un NaN : ∞/∞ ou $\sqrt{-1}$ ou $\infty - \infty$ ou $0/0$,
- drapeau division par zéro,
- drapeau débordement vers l'infini (*overflow*) quand le résultat d'une opération est trop grand en valeur absolue pour être représenté à cause de la borne supérieure des exposants représentables,
- drapeau débordement vers zéro (*underflow*) quand le résultat d'une opération est trop petit en valeur absolue pour être représenté à cause de la borne inférieure des exposants représentables.

Quatre modes d'arrondi sont définis : les arrondis dirigés (vers $-\infty$, vers $+\infty$, ou vers 0) et l'arrondi au plus proche. Ce dernier est ambigu quand le nombre à arrondir est équidistant de deux flottants, la norme impose de choisir celui dont la mantisse est paire (on parle d'arrondi pair).

Une fois le mode d'arrondi choisi, le résultat d'une opération est parfaitement spécifié, notamment le résultat est unique, on parle d'arrondi correct. La norme impose l'arrondi correct pour les quatre opérations de base (addition, soustraction, multiplication, division) ainsi que pour la racine carrée. On est donc sensé obtenir le même résultat sur des machines différentes respectant la norme ; mais souvent les machines utilisent des précisions intermédiaires étendues non-normalisées et les compilateurs s'autorisent à réorganiser les opérations (alors que les opérations flottantes perdent leur propriétés d'associativité), sans parler des problèmes de doubles arrondis (quand, en arrondi au plus proche, on utilise des précisions intermédiaires différentes).

Les arrondis vers l'infini permettent une arithmétique d'intervalle (toujours majorer ou minorer un résultat).

La norme ne spécifie rien sur les autres opérations (puissance, exponentielle, logarithme, sinus, cosinus, etc).

1.1.3 Le calcul d'erreurs

Comme la propagation d'erreur d'arrondi est inévitable, l'objectif est simplement de contrôler l'erreur. On introduit pour cela la notion d'ulp (*unit last place*) qui est le plus petit flottant représentable sur le squelette d'un nombre donné (c'est le poids du plus petit bit de la mantisse d'un flottant) : si $x = 1.b_{-1}...b_{1-p}\ 2^e$, alors $\mathrm{ulp}(x) = 2^{e+1-p}$.

La façon traditionnelle d'appréhender les erreurs consiste à introduire une erreur d'arrondi u (u est de l'ordre de 10^{-16} en double précision). Soit R une fonction d'arrondi et x le résultat d'un calcul ($+,*,/,-,\sqrt{}$) qui n'a pas explosé, la norme garantit :

$$\left|\frac{x - R(x)}{R(x)}\right| \leqslant u, \qquad \left|\frac{x - R(x)}{x}\right| \leqslant \frac{u}{1-u} \simeq u \tag{1.2}$$

La technique d'estimation d'erreur consiste à chaque opération à insérer un facteur $(1+\delta)$ avec $|\delta| \leqslant u$ et de voir comment les erreurs se propagent et se cumulent.

Exemple 1.1.1 ($(x+y)/2$)**.** On va introduire δ_1 pour la division par 2 et δ_2 pour l'addition

$$\begin{aligned} R((x+y)/2) &= R(x+y)/2 * (1+\delta_1) = (x+y)(1+\delta_2)(1+\delta_1)/2 \\ R((x+y)/2) - (x+y)/2 &= (\delta_1+\delta_2+\delta_1\delta_2)(x+y)/2 \\ \left|\frac{R((x+y)/2)-(x+y)/2}{(x+y)/2}\right| &\leqslant 2u \end{aligned} \tag{1.3}$$

On néglige l'ordre 2. On voit que l'erreur est bornée par $2u$.

Exemple 1.1.2 ((a^2-b^2) – effet de cancellation)**.** On a ici deux façon de calculer.

$$\begin{aligned} R(a^2-b^2) &= (a^2(1+\delta_1)-b^2(1+\delta_2))(1+\delta_3) \\ \left|R(a^2-b^2)-(a^2-b^2)\right| &\leqslant 2u(a^2+b^2) \\ \left|\frac{R(a^2-b^2)-(a^2-b^2)}{a^2-b^2}\right| &\leqslant 2u\frac{a^2+b^2}{|a^2-b^2|} \end{aligned} \tag{1.4}$$

Le + est dû au fait que les δ peuvent être négatifs.

Le facteur $\frac{a^2+b^2}{|a^2-b^2|}$ est le conditionnement de l'opération. On voit que l'erreur peut exploser si a et b sont proches. Par contre si on modifie la façon de calculer :

$$\begin{aligned} R((a-b)(a+b)) = R(a-b)R(a+b)(1+\delta_1) &= (a-b)(1+\delta_2)(a+b)(1+\delta_3)(1+\delta_1) \\ \left|\frac{R((a-b)(a+b))-(a-b)(a+b)}{(a-b)(a+b)}\right| &\leqslant 3u \end{aligned} \tag{1.5}$$

Et le calcul devient inconditionnellement précis.

L'exemple précédent présente le phénomène de cancellation (voir plus loin). Il montre aussi que les calculs ne sont plus associatifs ni distributifs et que le programmeur a un rôle dans la précision de ses calculs.

On illustre plus précisément quelques grandes sources d'erreur. On ne parle pas des erreurs de dépassement de capacité de représentation des nombres.

Cancellation

Il s'agit du principal problème du calcul en virgule flottante qui apparaît quand on fait la différence entre deux nombres très proches, la partie significative de leur différence après arrondi est alors très faible.

Un exemple flagrant est le calcul de e^{-a} avec $a > 0$. Si on utilise une série de Taylor, alors pour a grand le calcul peut être faux quelque soit la troncature de la série. Par contre si on prend l'inverse de e^a alors on peut obtenir un résultat « exact ».

Un autre exemple est l'évaluation de la fonction suivante :

$$f(x) = \frac{1-\cos(x)}{x^2} \text{ en } x = 1,2\,10^{-5}$$

si on prend une précision à 10 digits décimaux, on a :

$$\begin{aligned} \cos(x) &= 0,9999999999 \\ 1-\cos(x) &= 0,0000000001 \\ f(x) = \frac{10^{-10}}{1,44\,10^{-10}} &= 0,6944 \end{aligned}$$

alors que l'on sait que $\forall x \neq 0,\ 0 \leqslant f(x) \leqslant 0,5$.

La précision à 10 digits n'a pas permis d'avoir 1 digit juste sur le résultat. Le problème vient du fait que $1-\cos(x)$ n'a qu'un seul chiffre significatif alors que la soustraction est exacte (l'approximation vient du cos). La soustraction fait donc ressortir une approximation précédente. Si à la place on avait écrit :

$$f(x) = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2$$

alors on aurait obtenu $f(x) = 0,5$ qui est exact à 10 digits.

Sommation et ordonnancement

On rappelle que $\sum_{i=1}^{n} \frac{1}{i} - \ln(n) \to \gamma$ où γ est la constante d'Euler. γ est un nombre mal connu, on a $\gamma \simeq 0.5772156649\ldots$ (10 premières décimales exactes).

On va supposer γ et ln suffisamment précises. On évalue l'effet de la sommation (par exemple en précision simple) et en particulier de l'ordonnancement. On donne dans le code 1.1 4 façon de calculer.

La première fait la somme dans l'ordre ascendant. Au bout d'un certain temps $s \gg 1/i$ et on perd beaucoup de chiffres significatifs lors de la sommation. La seconde fait la somme en commençant par les petits nombres, ça évite d'avoir une somme intermédiaire beaucoup plus grande que les termes à ajouter. La troisième est de type *divide and conquer* avec mise en œuvre récursive. La dernière méthode (de Kahan), plus évoluée essaie d'estimer l'erreur pour la compenser. On donne l'évolution de l'erreur en fonction de n dans la figure 1.2.

En simple précision les problèmes arrivent à partir de $n = 10^4$, en double précision 10^{12}.

Code 1.1 – 4 façons de calculer la constante d'Euler (Octave)

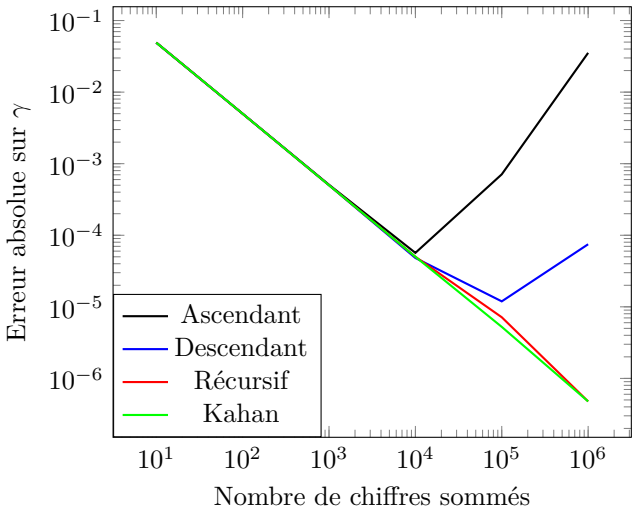
```
Euler = 0.5772156649; % on suppose n donné
s = 0; for i = 1 : 10^n; % Version ascendante
s = single(s + single(1/i));
endfor
disp(["Erreur asc. : ",num2str(s-log(10^n)-Euler)]);

s = 0; for i = 1 : 10^n; % Version descendante
s = single(s + single(1/(10^n+1-i)));
endfor
disp(["Erreur des.: ",num2str(s-log(10^n)-Euler)]);

function s = recu(d,f); % Version recursive
  if d==f; s=single(1/d);
  else s = single(recu(d,floor((d+f)/2)) + recu(floor((d+f)/2)+1,f));
  endif
endfunction
disp(["Erreur rec.: ",num2str(recu(1,10^n)-log(10^n)-Euler)]);

compensation = 0.0 ; s=0;
for i = 1:10^n ; % Version Kahan
toadd = single(1 / i) - compensation ;
sTemp = single(s + toadd);
compensation = single(single(sTemp - s)  - toadd) ;
s = sTemp;
endfor
disp(["Erreur Kah.: ",num2str(s-log(10^n)-Euler)]);
```

1.2 Stockage des données matricielles

On s'intéresse ici à la question du stockage d'une matrice dans la mémoire d'un ordinateur. Les matrices sont des objets bidimensionnels alors que la mémoire d'un ordinateur est un objet 1D où la contiguïté et l'alignement des données est primordial pour tirer les meilleures performances. Suivant l'utilisation, différentes mesures sont utiles : compacité du stockage, facilité pour accéder aux données, facilité pour insérer des données, facilité pour redimensionner.

Stockage dense

C'est un stockage où toutes les données sont stockées, y compris les zéros. C'est un stockage gourmand en mémoire mais très efficace par ailleurs. Comme la mémoire est réservée, il est facile d'insérer des données ; il est facile d'accéder aux données qui sont bien alignées ; pour le calcul les optimisations sont faciles. Cela peut



Remarque : pour 10^6 chiffres sommés tous les algorithmes sauf Kahan donnent une erreur négative (dans les autres cas l'erreur était positive).

FIGURE 1.2 – Erreur de sommation

être observé sur les temps de calculs du code 1.2 où des opérations semblant similaires mettent des temps très différents pour s'exécuter.

On distingue deux types de stockage : par ligne ou par colonne. Pour une matrice $m \times n$, avec la convention de numérotation C (les indices commencent par 0), m_{ij} est stocké à la position $(j + i * n)$ en stockage ligne et $(i + j * m)$ en stockage colonne. La dimension (n en ligne, m en colonne) utilisée pour repérer les coefficients est appelée *leading dimension* (LDA dans les bibliothèques BLAS et LAPACK). Sous réserve que suffisamment de mémoire a été allouée, il est facile d'ajouter des colonnes en stockage colonne, par contre ajouter des lignes réclame des déplacements mémoire.

Le stockage colonne est souvent le défaut en fortran, matlab, octave, le stockage ligne est classique en C. Les bibliothèques historiques pour la manipulation de matrices pleines sont BLAS et LAPACK, une bibliothèque plus moderne est EIGEN3.

```
n=10^4;
a = rand(n);%matrice n x n

b = zeros(n,1); % Columns
tic
for ii = 1:n ; b = b + a(:,ii); endfor
toc
```

```
b = zeros(1,n);% Rows
tic
for ii = 1:n; b = b + a(ii,:); endfor
toc

c = ones (n,1);
tic ; c= a*c; toc
tic ; c= (a'*c)'; toc
```

Code 1.2 – Effet du stockage sur la vitesse des opérations (octave)

Stockage creux

Le stockage creux (*sparse* en anglais) vise à éviter de stocker tous les zéros d'une matrice. Il est indispensable pour les grandes matrices éléments/volumes/différences-fini(e)s qui sont très peu peuplées. Une information utile est le NNZ (*number of non-zeros*) à comparer avec la taille d'un stockage plein mn.

On distingue différents types de stockage :

— Le stockage AIJ : on stocke 2 tableaux d'entiers et un tableau de réels tous de taille NNZ : I, J, $A(k) = a_{I(k)J(k)}$. De manière à faciliter la modification de valeurs, on autorise les tableaux à dépasser NNZ, avec la convention que les coefficients s'additionnent sur les emplacements répétés.

— Le stockage par ligne (resp. colonne) : on décrit la matrice ligne par ligne, on n'a plus besoin de stocker dans le tableau I (de taille m) que le nombre d'indices de la ligne.

— Le stockage par ligne (resp. colonne) compressée : si on sait que la ligne contient des blocs contigus de données, on décrit la ligne comme une succession de blocs (première colonne, nombre de coeff dans le bloc), ça permet de réduire la taille de J ; I stocke le nombre de blocs par ligne.

— Le stockage ligne de ciel (*skyline*) par ligne (resp. colonne) : si on sait que la matrice contient peu de zéros dans sa bande, on stocke un bloc par ligne. On économise beaucoup sur J au prix d'un peu plus de stockage sur A.

On rappelle au passage la notion de bande d'une matrice : il s'agit d'un tableau de $2m$ entiers qui pour chaque ligne contient l'indice de la première et de la dernière colonne non-nulle. Bien sûr si jamais une matrice est symétrique, on ne stocke que la moitié des coefficients. Il arrive aussi fréquemment que le profil de la matrice soit symétrique mais pas nécessairement les coefficients A.

Manipuler une matrice creuse est extrêmement pénible, et il est recommandé de faire appel à des bibliothèques extérieures et de bien étudier les méthodes proposées. Par exemple dans Matlab, un bon usage de la méthode `sparse()` permet d'économiser des facteurs de temps colossaux.

Numéroration

La numérotation joue une rôle important dans les stockages compressés ou skyline en limitant le nombre de zéros à l'intérieur de la bande. En anticipant un peu, elle joue aussi un rôle très important dans le calcul où de nombreuses factorisations préservent la bande (mais peuplent densément la bande).

Par exemple pour une factorisation de Cholesky $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ on a :

$$\mathbf{A} = \begin{pmatrix} 1. & 0. & 0. & 0. & 1. \\ 0. & 1. & 0. & 0. & 1. \\ 0. & 0. & 1. & 0. & 1. \\ 0. & 0. & 0. & 1. & 1. \\ 1. & 1. & 1. & 1. & 5. \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 1. & 0. & 0. & 0. & 0. \\ 0. & 1. & 0. & 0. & 0. \\ 0. & 0. & 1. & 0. & 0. \\ 0. & 0. & 0. & 1. & 0. \\ 1. & 1. & 1. & 1. & 1. \end{pmatrix} \tag{1.6}$$

Si on permute le ddl 1 avec le ddl 5, on obtient

$$\mathbf{A} = \begin{pmatrix} 5. & 1. & 1. & 1. & 1. \\ 1. & 1. & 0. & 0. & 0. \\ 1. & 0. & 1. & 0. & 0. \\ 1. & 0. & 0. & 1. & 0. \\ 1. & 0. & 0. & 0. & 1. \end{pmatrix}, \quad \mathbf{L} = \begin{pmatrix} 2.236068 & 0. & 0. & 0. & 0. \\ 0.4472136 & 0.8944272 & 0. & 0. & 0. \\ 0.4472136 & -0.2236068 & 0.8660254 & 0. & 0. \\ 0.4472136 & -0.2236068 & -0.2886751 & 0.8164966 & 0. \\ 0.4472136 & -0.2236068 & -0.2886751 & -0.4082483 & 0.7071068 \end{pmatrix} \tag{1.7}$$

La taille de bande (symétrique) de la première matrice était $(1, 1, 1, 1, 5)$ et pour la seconde $(1, 2, 3, 4, 5)$.

Là encore, les bibliothèques modernes incluent une analyse préalable du profil de la matrice (analyse symbolique, sans faire appel à la valeur des coefficients) qui permet de la renuméroter et d'ordonner les opérations de manière efficace.

Stockage de rang faible

Les matrices de rang faible jouent un rôle très important dans de nombreuses méthodes modernes car elles conduisent à des calculs peu chers (cf. Sherman Morrisson un peu plus tard) et bien alignés.

Soit A une matrice $m \times n$, si A est de rang r, il existe U $m \times r$ et V $n \times r$ tels que $A = UV^T$. À noter que cette représentation n'est pas unique et qu'il faut rajouter des conditions pour la spécifier complètement (voir la SVD plus loin).

AU lieu de stocker les mn coefficients de A qui est très probablement dense, on stocke les $r(m+n)$ coefficients de U et V, ce qui est très efficace si $r \ll \min(m, n)$ (définition du rang faible).

Il est alors important de comprendre que les opérations sur A se font alors en fait sur U et V par exemple, si x est un vecteur, quand on écrit Ax en fait, on code $U * (V^T * x)$ et le nombre d'opération est proportionnel à r.

Remarque. C'est peut-être l'occasion de rappeler qu'il faut se méfier des logiciels quand il s'agit d'exécuter plus de deux opérations : si vous écrivez $U * V^T * x$, un logiciel comme Matlab exécute la ligne de gauche à droite : $(U * V^T) * x$, ce qui est très mauvais par rapport à $U * (V^T * x)$. À moins d'utiliser un logiciel très intelligent qui prend le temps avant de faire des choix (évaluation paresseuse), il vaut mieux réfléchir et spécifier clairement l'ordre des opérations, dans la majorité des cas l'utilisateur sait ce qui est bon pour lui.

1.2.1 Stockage distribué

On s'intéresse à la distribution des données sur plusieurs unités de calcul indépendantes. Cela correspond assez naturellement au stockage par blocs d'une matrice. On rencontre deux situations.

Dans la première situation, la matrice creuse $\mathbf{A}$ apparaît naturellement sous une forme additive $\mathbf{A} = \sum \mathbf{J}^s \mathbf{A}^s \mathbf{J}^{s^T}$ où $\mathbf{A}^s$ est appelée contribution locale et $\mathbf{J}^s$ est une matrice booléenne qui fait le transfert entre degré de liberté local et degré de liberté global. Cette situation est rencontrée naturellement dans les méthodes de décomposition de domaine de type BDD et FETI. Il suffit alors de stocker $\mathbf{A}^s$ par processeur ainsi que les relations de connectivité associées à $\mathbf{J}^s$ car un processeur parle à peu de voisins (grace au profil creux de la matrice).

Dans le deuxième cas, la matrice $\mathbf{A}$ n'est pas donnée sous forme additive, c'est la situation qui vient naturellement quand on utilise des méthodes de décomposition de domaine avec recouvrement. On choisit alors d'attribuer des lignes (ou des colonnes) aux processeurs. En général l'orientation du stockage est choisie en accord avec le stockage local des données (distribution par lignes d'un stockage par ligne) pour faciliter les mouvements entre données.

Quant au stockage des vecteurs, il est naturel d'attribuer des lignes aux processeurs. Cependant, pour éviter les complications, il est souvent utile de stocker les mêmes lignes sur des processeurs différents. Par exemple une interface ou un overlap communs sont stockés par les deux processeurs voisins. Suivant les cas, on peut stocker les composantes du vecteur assemblé $\mathbf{J}^{s^T}\mathbf{v}$ où les composantes avant assemblage ($\mathbf{v}^s$) telles que $\mathbf{v} = \sum \mathbf{J}^s \mathbf{v}^s$.

1.3 Complexité des opérations

1.3.1 Fast Matrix Multiplication

Il s'agit ici simplement de montrer qu'il est possible d'améliorer des méthodes même très basiques.

Soit $\mathbf{A}$ et $\mathbf{B}$ deux matrices $2n \times 2n$. On veut calculer leur produit $\mathbf{C} = \mathbf{AB}$. On partitionne les matrices en blocs $n \times n$.

L'algorithme naïf est :

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{1,2} &= \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\
\mathbf{C}_{2,1} &= \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} \\
\mathbf{C}_{2,2} &= \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2}
\end{aligned} \tag{1.8}$$

Cela conduit à 8 multiplications, la complexité est en $O(n^3)$. La précision est en $nu|A||B|$ sur chaque composante.

L'algorithme de Strassen consiste à calculer :

$$\begin{aligned}
\mathbf{M}_1 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \\
\mathbf{M}_2 &:= (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1} \\
\mathbf{M}_3 &:= \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \\
\mathbf{M}_4 &:= \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1}) \\
\mathbf{M}_5 &:= (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \\
\mathbf{M}_6 &:= (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2}) \\
\mathbf{M}_7 &:= (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2})
\end{aligned} \tag{1.9}$$

il n'y a que 7 multiplications. On a finalement :

$$\begin{aligned}
\mathbf{C}_{1,1} &= \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 \\
\mathbf{C}_{1,2} &= \mathbf{M}_3 + \mathbf{M}_5 \\
\mathbf{C}_{2,1} &= \mathbf{M}_2 + \mathbf{M}_4 \\
\mathbf{C}_{2,2} &= \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6
\end{aligned} \tag{1.10}$$

Cet algorithme conduit à une complexité de $O(n^{2,807})$, par ailleurs, il se met en œuvre efficacement (bonne utilisation du cache). Sur cet algorithme brut, la précision est inférieure à la version naïve.

L'algorithme de Strassen n'est pas le plus performant, en fait la complexité optimale n'est pas connue. Actuellement, plusieurs algorithmes conduisent à $O(n^{2,3})$ et des précisions quasi optimales.

1.3.2 Opérations sur données distribuées

On étudie ici rapidement la complexité ajoutée par la distribution des données sur des opérations de base.

Supposons la situation d'une matrice à structure additive $\mathbf{A} = \sum \mathbf{J}^s \mathbf{A}^s \mathbf{J}^{s^T}$ que l'on veut multiplier avec un vecteur $\mathbf{v}$ assemblé et stocké sous la forme $\mathbf{J}^{s^T}\mathbf{v}$. On a :

$$\mathbf{w} = \mathbf{A}\mathbf{v} = \sum \mathbf{J}^s \mathbf{A}^s (\mathbf{J}^{s^T}\mathbf{v}) \tag{1.11}$$

On lisant de droite à gauche, on voit qu'il faut dans un premier temps appliquer le produit local $\mathbf{w}^s = \mathbf{A}^s(\mathbf{J}^{s^T}\mathbf{v})$ en parallèle, suivi de l'assemblage $\mathbf{w} = \sum \mathbf{J}^s \mathbf{w}^s$. Cette opération requiert de communiquer entre les processeurs. Plus précisément il s'agit d'une communication entre voisins, tous les voisins s'échangent des données. On parle d'opération collective creuse (NEIGHBOR_ALLTOALL en protocole MPI). Cette opération supporte assez bien le passage au grand nombre de processeurs car le nombre de maximal de voisins est en pratique limité (indépendant du nombre total de sous-domaine).

On s'intéresse maintenant au produit scalaire. Soit à calculer $\mathbf{v}^T\mathbf{w}$. Le cas favorable est celui d'un vecteur assemblé stocké sous la forme $\mathbf{J}^{s^T}\mathbf{v}$ et celui d'un vecteur stocké non-assemblé $\mathbf{w} = \sum \mathbf{J}^s \mathbf{w}^s$ en effet dans ce cas, on a :

$$\mathbf{v}^T\mathbf{w} = \mathbf{v}^T(\sum \mathbf{J}^s \mathbf{w}^s) = \sum (\mathbf{J}^{s^T}\mathbf{v})^T \mathbf{w}^s \tag{1.12}$$

On obtient une somme simple de calculs locaux. Dans le cas où les deux vecteurs sont assemblés, il faut trouver un mécanisme pour éviter des sommer les redondances. Dans ce cas, on peut considérer le « désassemblage » $\mathbf{w}^s = \mathbf{J}^{s^\dagger}\mathbf{w}$. La pseudo-inverse de $\mathbf{J}^s$ est souvent appelée opérateur de *scaling*.

In fine, on se retrouve à sommer des scalaires obtenus par des calculs indépendants par processeur. Le résultat (unique) de cette comme doit être connu de tous les processeurs. On parle de réduction globale par sommation (ALL_REDUCE(...,MPI_SUM) en protocole mpi). C'est une opération dont la complexité est en $\log_2(N)$ où N est le nombre de processeurs, les performances sont d'ailleurs légèrement moins bonnes quand N n'est pas une puissance de 2. Pour les très grands nombres de sous-domaines $\log_2(N)$ peut ne pas être négligeable et causer des goulots d'étranglement. Il est recommandé de minimiser les appels à ces méthodes voire à les masquer en les réalisant en toile de fond pendant que d'autres opérations ne nécessitant pas le résultat final se déroulent (*communication hiding algorithms*).

1.4 Compilateurs et interpréteurs

Quand on programme, on a plusieurs choix de plateformes, avec leurs avantages et inconvénients.

Les langages interprétés (anciennes version de Matlab, Octave, Python, Scilab,...) fonctionnent sur le principe du REPL (Read, Execute, Print, Loop). Ils ont l'avantage de donner un accès facile au contenu de la mémoire mais ils n'ont pas d'opportunité pour s'optimiser car ils exécutent une ligne après l'autre sans pouvoir anticiper. Il faut donc programmer en utilisant des fonctions préprogrammées (souvent compilées) de haut niveau. Il faut donc chercher le nom des méthodes existantes. Il est également important d'avoir une gestion de la mémoire intelligente et ne pas tomber dans les pièges de la facilité cf code 1.3. La dépendance à l'interpréteur qui n'est pas nécessairement disponible sur toutes les machines peut aussi être un problème.

```
N = 10000; M=1000;

c = rand(M,1) ;
tic
for i = 1:100
c = [c, rand(M,1)];
endfor
toc
```

```
C = zeros(M,N);
tic
for i = 1:101
c(:,i) = rand(M,1);
endfor
toc
```

Code 1.3 – Réallocation vs allocation une fois pour toute (octave)

À l'autre bout du spectre, les langages compilés (c, c++, fortran) consistent à donner des blocs de codes à un compilateur qui va pouvoir en optimiser l'exécution (parfois au prix d'une compilation lente). Il est utile de donner le maximum d'information au compilateur pour qu'il puisse au mieux s'optimiser. Pour coder de manière générique tout en optimisant le code, on peut utiliser des *templates* en c++ ou d'autres techniques. Typiquement il est intéressant de compiler des versions 2D et 3D des codes avec la dimension de l'espace donnée

```cpp
#include <ctime>
#include <iostream>
#include <array>
#include <vector>

int main(int argc, char *argv[]) {
double dt;
std::clock_t begin,end;
unsigned nb=10000;//number of nodes

// Version brute
begin = clock();
std::vector< std::vector< double > > b1 ;
b1.resize(nb);
for (int i = 0; i<nb;++i) {
b1[i].resize(3);
for (int j = 0; j<3; ++j){
b1[i][j]=i+j;}}
end = clock();
dt = double(end-begin)/CLOCKS_PER_SEC;
std::cout << "Time vector \t" << dt << std::endl;

// Exploitation de la dimension fixe
begin = clock();
std::vector< std::array< double,3 > > b2 ;
b2.resize(nb);
for (int i = 0; i<nb;++i) {
for (int j = 0; j<3; ++j){
b2[i][j]=i+j;}}
end = clock();
dt = double(end-begin)/CLOCKS_PER_SEC;
std::cout << "Time array \t" << dt << std::endl;
}
```

Code 1.4 – Spécification de la dimension fixe de l'espace (c++)

« en dur », voir le code 1.4 où l'on exploite le fait que tous les noeuds ont le même nombre de coordonnées ou pas.

Le problème des codes compilés est l'accès aux données en cours d'exécution qui facilite le debogage, et les problèmes d'allocation mémoire. Pour cela, il est bon de savoir utiliser un débogeur (gdb) et un vérificateur de mémoire (valgrind). La dépendance aux librairies extérieures peut compliquer la compilation.

Un intermédiaire est l'utilisation de langage compilés à la volée ou *just in time* (julia, matlab nouvelle version, python+numba, c++ avec llvm-jit). À la première exécution d'une fonction, elle est compilée, ce qui permet de l'optimiser et rendra rapide toutes les exécutions suivantes.

Chapitre 2

Algèbre

2.1 Prérequis et notations

2.1.1 Notations

On note en minuscule souligné les vecteurs, en calligraphié les opérateurs linéaires. On note en gras les grandeurs exprimées dans une base : en majuscule les matrices, en minuscule les vecteurs. $\mathbf{A}$ est la matrice de coefficients (a_{ij}). $\mathbf{A}$ est la représentation de $\mathcal{A}$ dans une base (sauf précision contraire, la base canonique) tout comme $\mathbf{v}$ est la représentation du vecteur $\underline{v}$, ses composantes sont les (v_i). On passera implicitement d'une famille de vecteurs colonnes $(\mathbf{a}_i)$ à une matrice : $\mathbf{A} = [\mathbf{a}_1, \dots \mathbf{a}_n]$.

Par défaut l'espace vectoriel $\mathbb{E}$ sur le corps $\mathbb{K}$ ($\mathbb{R}$ ou $\mathbb{C}$) est de dimension n, les matrices sont de dimension $m \times n$ (m lignes, n colonnes). Les bornes des indices sont omises quand il n'y a pas de risque de confusion. Les vecteurs $(\mathbf{e}_i)$ forment la base orthonormale canonique ($e_{ij} = \delta_{ij}$ symbole de Kronecker).

En général, les propriétés sont énoncées dans $\mathbb{C}$ et il est précisé si elles restent valables dans $\mathbb{R}$.

2.1.2 Rappels d'algèbre linéaire

Les notions suivantes sont supposées acquises :

— espace vectoriel $\mathbb{E}$ sur un corps $\mathbb{K}$

— sous-espaces vectoriels, sommes directes, sous-espaces supplémentaires,

— famille libre, liée, génératrice, base, sous-espace engendré (vect).

— dimension finie, théorème de la base incomplète,

— application linéaire, image, rang, noyau (à gauche et à droite), endo/iso/automorphisme,

— théorème du rang $\mathcal{A} : \mathbb{E} \to \mathbb{F}$, $\mathrm{rg}(\mathcal{A}) + \dim(\ker(\mathcal{A})) = \dim(\mathbb{E})$

— matrice, opérations de base, décomposition en blocs.

> Une matrice $\mathbf{A}$ est la représentation d'une application linéaire $\mathcal{A}$ dans des bases des espaces de départ et d'arrivée données. Le produit $\mathbf{Av}$ est la représentation du vecteur $\mathcal{A}(\underline{v})$ dans la base d'arrivée. D'un point de vue pratique, il est utile d'interpréter ce produit comme une combinaison linéaire des colonnes de $\mathbf{A}$: $\mathbf{w} = \mathbf{Av} = \sum_j \mathbf{a}_j v_j$. De même le produit $\mathbf{C} = \mathbf{AB}$ de deux matrices est la représentation de la composition de deux applications linéaires, d'un point de vue pratique chaque colonne de $\mathbf{C}$ est la combinaison de colonnes de $\mathbf{A}$ de coefficients donnés par une colonne de $\mathbf{B}$: $\mathbf{c}_j = \sum_k \mathbf{a}_k b_{kj}$, chaque ligne de $\mathbf{C}$ est la combinaison de lignes de $\mathbf{B}$ de coefficients donnés par une ligne de $\mathbf{A}$. Ce genre d'interprétation permet de comprendre rapidement l'effet de multiplications de matrices particulières à gauche ou à droite.

— matrice identité $\mathbf{I}$, inverse de matrice, déterminant, trace,

— changement de base.

> Une application linéaire $\mathcal{A} : \mathbb{E} \to \mathbb{E}$ est représentée par une matrice $\mathbf{A}$ dans une base $(\underline{e}_i)$ et par une matrice $\mathbf{B}$ dans une base $(\underline{f}_i)$. Les matrices $\mathbf{A}$ et $\mathbf{B}$ sont dites *semblables*. Si $\mathbf{P}$ est la *matrice de passage* de la base $(\underline{e}_i)$ dans la base $(\underline{f}_i)$ (la j-ème colonne de $\mathbf{P}$ est le vecteur $\underline{f}_j$ décomposé dans la base $(\underline{e}_i)$: $\underline{f}_j = \sum_i p_{ij} \underline{e}_i$), on a :
>
> $$\mathbf{B} = \mathbf{P}^{-1}\mathbf{AP} \tag{2.1}$$

— matrice diagonale, triangulaire, groupe (pour la multiplication) des matrices triangulaires inversibles,
— norme, topologie des espaces de dimension finie,

Une norme est une forme de $\mathbb{E}$ dans $\mathbb{R}$ qui vérifie les propriétés suivantes.

$$\begin{aligned}
\forall \underline{v} \in \mathbb{E},\ \|\underline{v}\| \geqslant 0 &\quad \text{forme positive} \\
\forall (\underline{v}, \alpha) \in \mathbb{E} \times \mathbb{K},\ \|\alpha \underline{v}\| = |\alpha| \|\underline{v}\| &\quad \text{forme homogène} \\
\|\underline{v}\| = 0 \Leftrightarrow \underline{v} = 0 &\quad \text{forme définie} \\
\forall (\underline{u}, \underline{v}) \in \mathbb{E}^2,\ \|\underline{u} + \underline{v}\| \leqslant \|\underline{u}\| + \|\underline{v}\| &\quad \text{inégalité triangulaire (de Minkovski)}
\end{aligned} \tag{2.2}$$

Deux normes $\|.\|_1$ et $\|.\|_2$ sont équivalentes s'il existe deux réels α_1 et α_2 tels que $\forall \underline{v} \in \mathbb{E},\ \alpha_1 \|\underline{v}\|_1 \leqslant \|\underline{v}\|_2 \leqslant \alpha_2 \|\underline{v}\|_1$. Deux normes équivalentes définissent la même topologie.
En dimension finie toutes les normes sont équivalentes, toutes les applications linéaires sont continues. Il n'y a qu'une seule notion de convergence (forte / faible).

— produit scalaire hermitien (euclidien), orthogonalité, transposition, adjoint,

Un produit scalaire hermitien (euclidien) est une *forme sesquilinéaire* (*forme bilinéaire*) de $\mathbb{E} \times \mathbb{E}$ dans $\mathbb{C}$ (dans $\mathbb{R}$) qui vérifie :

$$\begin{aligned}
\forall (\underline{u}, \underline{u}', \underline{v}) \in \mathbb{E}^3,\ \lambda \in \mathbb{C},\ \langle \underline{u} + \lambda \underline{u}', \underline{v} \rangle = \langle \underline{u}, \underline{v} \rangle + \lambda \langle \underline{u}', \underline{v} \rangle &\quad \text{forme linéaire à gauche} \\
\forall (\underline{u}, \underline{v}, \underline{v}') \in \mathbb{E}^3,\ \lambda \in \mathbb{C},\ \langle \underline{u}, \underline{v} + \lambda \underline{v}' \rangle = \langle \underline{u}, \underline{v} \rangle + \bar{\lambda} \langle \underline{u}, \underline{v}' \rangle &\quad \text{forme antilinéaire à droite} \\
\forall (\underline{u}, \underline{v}) \in \mathbb{E}^2,\ \langle \underline{u}, \underline{v} \rangle = \overline{\langle \underline{v}, \underline{u} \rangle} &\quad \text{forme symétrique} \\
\forall \underline{u} \in \mathbb{E},\ \langle \underline{u}, \underline{u} \rangle \geqslant 0 &\quad \text{forme positive} \\
\langle \underline{u}, \underline{u} \rangle = 0 \Leftrightarrow \underline{u} = 0 &\quad \text{forme définie}
\end{aligned} \tag{2.3}$$

L'opération matricielle associée est la transposition-conjugaison $\langle \underline{u}, \underline{v} \rangle = \mathbf{u}^H \mathbf{v} = \bar{\mathbf{u}}^T \mathbf{v}$.
Chaque coefficient de $\mathbf{C} = \mathbf{A}^H \mathbf{B}$ s'interprète comme un produit scalaire de colonnes des deux matrices : $c_{ij} = \mathbf{a}_i^H \mathbf{b}_j$.

Théorème 2.1.1. $\mathrm{Im}(\mathbf{A^H}) \overset{\perp}{\oplus} \mathrm{Ker}(\mathbf{A}) = \mathbb{C}^n$

Démonstration. Soit $\mathbf{x} \in \mathrm{Ker}(\mathbf{A})$ donc $\mathbf{Ax} = 0$, et $\mathbf{y} \in \mathrm{Im}(\mathbf{A^H})$ donc $\exists \mathbf{z} \in \mathbb{E} /\ \mathbf{y} = \mathbf{A^H z}$, on a alors $\mathbf{x}^H \mathbf{y} = \mathbf{x}^H \mathbf{A^H z} = (\mathbf{Ax})^H \mathbf{z} = 0$, d'où l'orthogonalité et donc l'indépendance. Le théorème du rang permet de conclure. □

Remarque. Le théorème précédent s'étend à des orthogonalités au sens des matrices hermitiennes définies positives (HPD) : si $\mathbf{C}$ est HPD, on a $\mathrm{Im}(\mathbf{CA^H}) \overset{\perp}{\oplus} \mathrm{Ker}(\mathbf{AC}) = \mathbb{C}^n$ ou encore $\mathrm{Im}(\mathbf{A^H}) \overset{\perp_C}{\oplus} \mathrm{Ker}(\mathbf{AC}) = \mathbb{C}^n$.

— projection oblique et orthogonale,

Un projecteur $\mathbf{P}$ est un endomorphisme idempotent $\mathbf{P}^2 = \mathbf{P}$. Si $\mathbf{P}$ est un projecteur alors $(\mathbf{I} - \mathbf{P})$ et $\mathbf{P}^H$ sont des projecteurs. Un projecteur projette sur son image parallèlement à son noyau (ou orthogonalement à $\mathrm{Ker}(\mathbf{P})^\perp$) qui sont des sous-espaces supplémentaires. Si $\mathbf{V}$ est une base de l'image et $\mathbf{W}$ une base de l'orthogonal au noyau, on a :

$$\mathbf{P} = \mathbf{V}(\mathbf{W}^H \mathbf{V})^{-1} \mathbf{W}^H \tag{2.4}$$

Démonstration. On voit que $\mathbf{P}^2 = \mathbf{P}$ donc $\mathbf{P}$ est un projecteur. On voit clairement que les vecteurs de $\mathrm{Im}(\mathbf{V})$ sont invariants et ceux de $\mathrm{Im}(\mathbf{W})^\perp$ renvoient 0. □

Si $\mathbf{V} = \mathbf{W}$ le projecteur est orthogonal et alors $\mathbf{P}$ est hermitien. Si $\mathbf{P}$ est un projecteur orthogonal sur $\mathbb{V}$ alors
— $\|\mathbf{x}\|_2^2 = \|\mathbf{Px}\|_2^2 + \|(\mathbf{I} - \mathbf{P})\mathbf{x}\|_2^2$.
— $\|\mathbf{x}\|_2 \geqslant \|\mathbf{Px}\|_2$.
— $\|\mathbf{P}\|_2 = 1$
— $\mathbf{P}$ a deux valeurs propres : 0 (associée à son noyau) et 1 (associée à son image).
— $\forall \mathbf{x}$, $\min_{\mathbf{y} \in \mathbb{V}} \|\mathbf{x} - \mathbf{y}\|_2 = \|\mathbf{x} - \mathbf{Px}\|_2$

— analyse spectrale

Les valeurs propres sont les racines du polynôme caractéristique $\det(\mathbf{A} - \lambda\mathbf{I})$. À chaque valeur propre on peut associer au moins un vecteur propre à droite solution de $(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0$ et un vecteur propre à gauche solution de $\mathbf{v}^H(\mathbf{A} - \lambda\mathbf{I}) = 0$. Puisque $\mathbb{C}$ est algébriquement clos chaque matrice a au moins une valeur propre (dans $\mathbb{C}$).

Le spectre est l'ensemble des valeurs propres $\mathrm{sp}(\mathbf{A}) = \{\lambda_i\} \subset \mathbb{C}$, le rayon spectral est $\rho(\mathbf{A}) = \max(|\lambda_i|)$. Les matrices triangulaires ont pour valeurs propres leurs coefficients diagonaux. Polynômes annulateurs caractéristique et minimal, lien entre trace, déterminant et valeurs propres.

Théorème 2.1.2 (de Gershgorin)**.** *Les valeurs propres d'une matrice sont contenues dans l'union des n disques (dans le plan complexe) fermés centrés sur a_{ii} de rayon $\sum_{j\neq i} |a_{ij}|$. En appliquant le théorème à la matrice transposée, on trouve que les valeurs propres sont aussi contenues dans l'union des disques fermés centrés sur a_{ii} de rayon $\sum_{j\neq i} |a_{ji}|$.*

Démonstration. Soit λ une valeur propre et $\mathbf{x}$ un vecteur propre associé, on a $\forall i,\ (\lambda - a_{ii})x_i = \sum_{j\neq i} a_{ij}x_j$, en choisissant i_0 qui réalise le $\max |x_i|$ (qui est non nul puisqu'un vecteur propre ne peut être nul), on a

$$|\lambda - a_{i_0 i_0}| \leqslant \sum_{j\neq i_0} |a_{i_0 j}| \left|\frac{x_j}{x_{i_0}}\right| \leqslant \sum_{j\neq i_0} |a_{i_0 j}|$$

□

Définition 2.1.1 (Matrice symétrique, hermitienne, normale, unitaire, orthogonale.)**.** Une matrice $\mathbf{A}$ est :
- hermitienne si $\mathbf{A}^H = \mathbf{A}$;
- anti-hermitienne si $\mathbf{A}^H = -\mathbf{A}$;
- unitaire si $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A} = \mathbf{I}$;
- normale si $\mathbf{A}\mathbf{A}^H = \mathbf{A}^H\mathbf{A}$
- symétrique si $\mathbf{A}$ est réelle et $\mathbf{A} = \mathbf{A}^T$;
- orthogonale si $\mathbf{A}$ est réelle et $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{I}$;

Proposition 2.1.3. *Une matrice se décompose de manière unique en parties hermitienne et anti-hermitienne :*

$$\mathbf{A} = \frac{\mathbf{A} + \mathbf{A}^H}{2} + \frac{\mathbf{A} - \mathbf{A}^H}{2}$$

Définition 2.1.2. À une matrice carrée $\mathbf{A}$ on peut associer une forme quadratique : $\mathbf{v} \mapsto \mathbf{v}^H\mathbf{A}\mathbf{v}$.

Proposition 2.1.4.

Une matrice hermitienne définit une forme quadratique réelle.

Une matrice anti-hermitienne définit une forme quadratique purement imaginaire.

Une matrice antisymétrique définit une forme quadratique nulle sur $\mathbb{R}$

Démonstration. Soit $\mathbf{v} \in \mathbb{E}$, et $\mathbf{A}$ une matrice carrée $\mathbf{v}^H\mathbf{A}\mathbf{v} = \left(\mathbf{A}^H\mathbf{v}\right)^H \mathbf{v} = \overline{\mathbf{v}^H\mathbf{A}^H\mathbf{v}}$

- Si $\mathbf{A}$ est hermitienne alors $\mathbf{v}^H\mathbf{A}\mathbf{v} = \overline{\mathbf{v}^H\mathbf{A}\mathbf{v}}$ donc la forme est réelle
- Si $\mathbf{A}$ est anti-hermitienne alors $\mathbf{v}^H\mathbf{A}\mathbf{v} = -\overline{\mathbf{v}^H\mathbf{A}\mathbf{v}}$ donc la forme est purement imaginaire
- Si $\mathbf{A}$ est anti-symétrique alors $\mathbf{v}^T\mathbf{A}\mathbf{v} = -\mathbf{v}^T\mathbf{A}\mathbf{v}$ donc la forme est nulle. □

Définition 2.1.3 (Matrice positive, matrice définie, ordre partiel de Löwner)**.** Soit une matrice carrée $\mathbf{A}$.
- Elle est positive si et seulement si la forme quadratique qui lui est associée est réelle positive $\forall \mathbf{v} \in \mathbb{E},\ \mathbf{v}^H\mathbf{A}\mathbf{v} \geqslant 0$.
- Elle est définie si et seulement si $\mathbf{v}^H\mathbf{A}\mathbf{v} = 0 \Leftrightarrow \mathbf{v} = 0$.

On peut munir l'ensemble des matrices hermitiennes d'un ordre partiel (de Löwner) : $\mathbf{A} \leqslant \mathbf{B} \Leftrightarrow \forall \mathbf{x} \in \mathbb{E}, \mathbf{x}^H\mathbf{A}\mathbf{x} \leqslant \mathbf{x}^H\mathbf{B}\mathbf{x}$. La positivité d'une matrice au sens de Löwner correspond à la positivité de la forme quadratique.

Remarque. Il existe une autre notion d'ordre partiel pour les matrices réelles (pas nécessairement symétriques) : $\mathbf{A} \geqslant \mathbf{B} \Leftrightarrow \forall (i,j),\ a_{ij} \geqslant b_{ij}$. Une matrice positive est alors une matrice dont tous les coefficients sont positifs.

Définition 2.1.4 (Matrice à diagonale dominante.)**.** Une matrice $\mathbf{A}$ est à diagonale dominante si $\forall i,\ |a_{ii}| \geqslant \sum_{j\neq i} |a_{ij}|$.

2.2 Réduction des matrices

2.2.1 Résultats généraux de matrices équivalentes

La *réduction* consiste à écrire une application dans la base qui rend la matrice associée la plus « simple » possible.

Théorème 2.2.1. *Principaux résultats de réduction :*
- *Étant donné une matrice* carré $\mathbf{A}$*, il existe une matrice* unitaire $\mathbf{U}$ *telle que la matrice* $\mathbf{T} = \mathbf{U}^{-1}\mathbf{A}\mathbf{U}$ *soit* triangulaire *(théorème de trigonalisation de Schur).*
- *Étant donné une matrice* normale $\mathbf{A}$*, il existe une matrice* unitaire $\mathbf{U}$ *telle que la matrice* $\mathbf{D} = \mathbf{U}^{-1}\mathbf{A}\mathbf{U}$ *soit* diagonale*.*
- *Étant donné une matrice* hermitienne *(respectivement* symétrique*)* $\mathbf{A}$*, il existe une matrice* unitaire *(respectivement* orthogonale*)* $\mathbf{U}$ *telle que la matrice* $\mathbf{D} = \mathbf{U}^{-1}\mathbf{A}\mathbf{U}$ *soit* diagonale réelle*.*

Proposition 2.2.2.
- *Les éléments diagonaux de* $\mathbf{T}$ *ou* $\mathbf{D}$ *sont les valeurs propres de* $\mathbf{A}$*.*
- *Les valeurs propres d'une matrice unitaire ou orthogonale sont des nombres complexes de module* 1*.*

Remarque. Pour mémoire, la matrice semblable la plus simple pour une matrice quelconque s'écrit sous la forme de Jordan (bloc-diagonale avec des 1 sur la première surdiagonale).

2.2.2 Décomposition en valeurs singulières

Un problème plus général consiste à trouver l'expression d'une *matrice équivalente* particulièrement simple. Dans ce cas là l'application n'est plus nécessairement un endomorphisme, $\mathcal{A} : \mathbb{E} \to \mathbb{F}$, les matrices associées peuvent être rectangulaires et on introduit des changements de base dans $\mathbb{E}$ et $\mathbb{F}$. On a donc $\mathbf{Q}$ isomorphisme de $\mathbb{F}$, $\mathbf{P}$ isomorphisme de $\mathbb{E}$ et

$$\mathbf{B} = \mathbf{Q}\mathbf{A}\mathbf{P} \tag{2.5}$$

Théorème 2.2.3. *Décomposition en valeurs singulières (SVD) : si* $\mathbf{A}$ *est une matrice complexe* $m \times n$ *(respectivement réelle), il existe deux matrices unitaires (respectivement orthogonales)* $\mathbf{U}$ *(*$n \times n$*) et* $\mathbf{V}$ *(*$m \times m$*) telles que*

$$\mathbf{U}^H\mathbf{A}\mathbf{V} = \mathbf{\Sigma} \qquad \mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \sigma_m \\ \vdots & & & 0 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \quad \textit{matrice "diagonale" } m \times n \tag{2.6}$$

où l'on a choisi de représenter le cas $n > m$*. Les* σ_i *sont appelées valeurs singulières de* $\mathbf{A}$*, ce sont les racines carrées des valeurs propres de la matrice* $\mathbf{A}^H\mathbf{A}$*, elles sont nécessairement* réelles, positives ou nulles *; on peut choisir de les disposer dans n'importe quel ordre, pour un ordre donné la décomposition est unique. Par la suite on choisit l'ordre décroissant et alors* $\sigma_1 = \sqrt{\rho(\mathbf{A}^H\mathbf{A})}$*.*

Remarque. Le résultat s'écrit aussi :

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^H = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^H \tag{2.7}$$

ce qui décompose la matrice en la somme de $\mathrm{rg}(\mathbf{A})$ contributions de rang 1.

Remarque. Quitte à se répéter, la SVD d'une matrice réelle ne fait intervenir que des matrices réelles.

Proposition 2.2.4. *Si* $\sigma_1 \geqslant \cdots \geqslant \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$ *on a :*

$$\begin{aligned} \mathrm{rg}(\mathbf{A}) = \mathrm{rg}(\mathbf{A^H}) = r \qquad & \dim(\mathrm{Ker}(\mathbf{A})) = m - r,\ \dim(\mathrm{Ker}(\mathbf{A^H})) = n - r \\ \mathrm{Ker}(\mathbf{A}) = \mathrm{vect}(\mathbf{v}_{r+1}, \cdots, \mathbf{v}_m) \qquad & \mathrm{Ker}(\mathbf{A^H}) = \mathrm{vect}(\mathbf{u}_{r+1}, \cdots, \mathbf{u}_n) \\ \mathrm{Im}(\mathbf{A}) = \mathrm{vect}(\mathbf{u}_1, \cdots, \mathbf{u}_r) \qquad & \mathrm{Im}(\mathbf{A^H}) = \mathrm{vect}(\mathbf{v}_1, \cdots, \mathbf{v}_r) \end{aligned} \tag{2.8}$$

La SVD est donc une transformation qui apporte énormément d'information, elle est peut être obtenue par des algorithmes numériquement très stables mais très coûteux.

Une valeur singulière très faible (ou nulle) indique que la matrice n'est pas inversible en pratique (matrice de rang numérique déficient). Le résultat suivant connecte la SVD à la distance d'une matrice aux matrices de rang déficient (la norme 2 des matrices est étudiée dans la section 2.3).

Proposition 2.2.5.

$$\min_{\mathrm{rg}(\mathbf{B})=k} \|\mathbf{A}-\mathbf{B}\|_2 = \|\mathbf{A}-\mathbf{A}_k\|_2 = \sigma_{k+1} \qquad o\grave{u}\ \mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^H \tag{2.9}$$

Cette équation montre que la distance d'une matrice aux matrices de rang k est caractérisée par la $(k+1)$-ème valeur singulière : en particulier, il suffit d'une perturbation de l'ordre de $\sigma_n > 0$ pour rendre une matrice non-inversible. On voit aussi que $\mathbf{A}_k$ est le meilleur représentant de $\mathbf{A}$ de rang k (au sens de la norme euclidienne), cette propriété est à la base des techniques de compression de données et de réduction de modèle.

2.2.3 Quotient de Rayleigh des matrices symétriques et hermitiennes

Définition 2.2.1 (Quotient de Rayleigh.)**.** Soit $\mathbf{A}$ une matrice hermitienne, son quotient de Rayleigh est la fonction :

$$\mathbf{v} \neq 0 \in \mathbb{E}, \qquad R_{\mathbf{A}}(\mathbf{v}) = \frac{\mathbf{v}^H \mathbf{A}\mathbf{v}}{\mathbf{v}^H \mathbf{v}} \in \mathbb{R} \tag{2.10}$$

On remarque que $R_{\mathbf{A}}(\alpha \mathbf{v}) = R_{\mathbf{A}}(\mathbf{v})$, donc on peut travailler sur la sphère unité euclidienne ($\mathbf{v}^H\mathbf{v} = 1$).

Proposition 2.2.6. *Si les valeurs propres sont ordonnées $\lambda_1 \geqslant \cdots \geqslant \lambda_n$ et qu'on note $\mathbf{p}_i$ les vecteurs propres orthonormés (en norme 2) associés. Si $\mathbb{E}_k = \mathrm{vect}(\mathbf{p}_1, \cdots, \mathbf{p}_k)$, et $\mathcal{E}_k$ est l'ensemble des sous-espaces de dimension k, on a :*

- $\lambda_k = R_{\mathbf{A}}(\mathbf{p}_k)$
- $\lambda_k = \min_{\mathbf{v}\in\mathbb{E}_k} R_{\mathbf{A}}(\mathbf{v})$
- $\lambda_k = \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} R_{\mathbf{A}}(\mathbf{v})$
- $\lambda_k = \max_{\mathbb{V}\in\mathcal{E}_k} \min_{\mathbf{v}\in\mathbb{V}} R_{\mathbf{A}}(\mathbf{v})$
- $\lambda_k = \min_{\mathbb{V}\in\mathcal{E}_{k-1}} \max_{\mathbf{v}\perp\mathbb{V}} R_{\mathbf{A}}(\mathbf{v})$
- $\{R_{\mathbf{A}}(\mathbf{v}), \mathbf{v} \in \mathbb{E}\} = [\lambda_1, \lambda_n] \subset \mathbb{R}$

Théorème 2.2.7 (d'entrelacement de Cauchy)**.** *Si $\mathbf{H}$ est une matrice hermitienne de taille n et $\mathbf{A}$ une sous-matrice principale de $\mathbf{H}$ de taille r, alors les valeurs propres de A séparent celles de $\mathbf{H}$ et plus précisément pour $1 \leqslant i \leqslant r$:*

$$\lambda_i(\mathbf{H}) \geqslant \lambda_i(\mathbf{A}) \geqslant \lambda_{n-(r-i)}(\mathbf{H})$$

Proposition 2.2.8 (Inégalité de Kantorovitch)**.** *Si $\mathbf{A}$ est une matrice hermitienne définie positive de valeurs propres extrémales λ_1 et λ_n alors $\forall \mathbf{v} \in \mathbb{E}$*

$$1 \leqslant R_{\mathbf{A}}(\mathbf{v}) R_{\mathbf{A}^{-1}}(\mathbf{v}) \leqslant \frac{(\lambda_1+\lambda_n)^2}{4\lambda_1\lambda_n} \tag{2.11}$$

Démonstration. Si $\mathbf{v}$ est un vecteur unitaire, alors $R_{\mathbf{A}}(\mathbf{v})R_{\mathbf{A}^{-1}}(\mathbf{v}) = \sum \lambda_i v_i^2 \sum \frac{v_i^2}{\lambda_i}$ avec $\sum v_i^2 = 1$. Par convexité de la fonction $\lambda \mapsto \frac{1}{\lambda}$ on a $\sum \frac{v_i^2}{\lambda_i} \geqslant \frac{1}{\sum \lambda_i v_i^2}$. Par convexité de l'épigraphe (zone comprise au dessus de $\frac{1}{\lambda}$ et sous le segment $\left[(\lambda_1, \frac{1}{\lambda_1}), (\lambda_n, \frac{1}{\lambda_n})\right]$) on a $\sum \frac{v_i^2}{\lambda_i} \leqslant \frac{1}{\lambda_1} + \frac{\sum v_i^2\lambda_i - \lambda_1}{\lambda_n - \lambda_1}\left(\frac{1}{\lambda_n} - \frac{1}{\lambda_1}\right) = \frac{1}{\lambda_1\lambda_n}\left(\lambda_1 + \lambda_n - \sum v_i^2\lambda_i\right)$. En posant $\tilde{\lambda} = \sum v_i^2\lambda_i$, on voit que $1 \leqslant R_{\mathbf{A}}(\mathbf{v})R_{\mathbf{A}^{-1}}(\mathbf{v}) \leqslant \frac{\tilde{\lambda}}{\lambda_1\lambda_n}\left(\lambda_1 + \lambda_n - \tilde{\lambda}\right) \leqslant \frac{(\lambda_n+\lambda_1)^2}{4\lambda_n\lambda_1}$ □

2.3 Normes

2.3.1 Normes vectorielles

Définition 2.3.1 (p-norme)**.** La classe la plus utilisée de normes vectorielles est celle des p-normes :

$$\|\mathbf{x}\|_p = (|x_1|^p + \cdots + |x_n|^p)^{1/p} \tag{2.12}$$

Théorème 2.3.1 (Inégalité de Holder et de Cauchy-Schwartz)**.** *:*

$$\begin{aligned} |\mathbf{x}^H\mathbf{y}| &\leqslant \|\mathbf{x}\|_p\|\mathbf{y}\|_q \qquad \frac{1}{p}+\frac{1}{q}=1 \\ |\mathbf{x}^H\mathbf{y}| &\leqslant \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \end{aligned} \tag{2.13}$$

Proposition 2.3.2. *On sait que toutes les normes sont équivalentes, les encadrements suivants sont « optimaux » :*

$$\begin{aligned} \|\mathbf{x}\|_2 &\leqslant \|\mathbf{x}\|_1 \leqslant \sqrt{n}\|\mathbf{x}\|_2 \\ \|\mathbf{x}\|_\infty &\leqslant \|\mathbf{x}\|_2 \leqslant \sqrt{n}\|\mathbf{x}\|_\infty \\ \|\mathbf{x}\|_\infty &\leqslant \|\mathbf{x}\|_1 \leqslant n\|\mathbf{x}\|_\infty \end{aligned} \tag{2.14}$$

2.3.2 Normes matricielles

Définition 2.3.2 (Norme matricielle)**.** Une norme matricielle (ou d'algèbre) est une norme sur l'espace des matrices qui vérifie une propriété de sous-multiplicativité :

$$\|\mathbf{AB}\| \leqslant \|\mathbf{A}\|\|\mathbf{B}\| \tag{2.15}$$

Les normes matricielles sont de très loin les plus intéressantes.

Exemple 2.3.1 (Norme non-matricielle)**.** $\max_{ij} a_{ij}$ est une norme non-matricielle sur l'espace des matrices. C'est la norme ∞ appliquée au vecteur $m \times n$ des coefficients de $\mathbf{A}$ (autrement dit cette norme oublie qu'une matrice c'est bien plus qu'un tableau de coefficients).

Définition 2.3.3 (Norme subordonnée)**.** Une norme matricielle subordonnée est définie à partir d'une norme vectorielle :

$$\|\mathbf{A}\|_p = \sup_{\substack{\mathbf{x}\in\mathbb{C}^n \\ \mathbf{x}\neq 0}} \frac{\|\mathbf{Ax}\|_p}{\|\mathbf{x}\|_p} = \max_{\substack{\mathbf{x}\in\mathbb{C}^n \\ \|\mathbf{x}\|_p=1}} \|\mathbf{Ax}\|_p \tag{2.16}$$

autrement dit $\|\mathbf{Ax}\|_p \leqslant \|\mathbf{A}\|_p\|\mathbf{x}\|_p$, avec au moins un vecteur où il y a égalité.

Proposition 2.3.3. $\|\mathbf{I}\|_p = 1$

Remarque. La définition de la norme subordonnée utilise un vecteur variant dans $\mathbb{C}^n$ même pour une matrice réelle. Néanmoins, dans ce dernier cas, le vecteur qui réalise l'égalité est réel. Bien que ce ne soit pas le cas pour les normes usuelles (1,2, ∞), on peut construire une norme vectorielle et sa norme matricielle subordonnée dont le max est différent sur $\mathbb{C}^n$ et $\mathbb{R}^n$.

Les normes 1 et ∞ sont facilement calculables :

$$\|\mathbf{A}\|_1 = \max_{1\leqslant j\leqslant n} \sum_{i=1}^{m} |a_{ij}| \qquad \|\mathbf{A}\|_\infty = \max_{1\leqslant i\leqslant m} \sum_{j=1}^{n} |a_{ij}| \qquad \|\mathbf{A}\|_1 = \|\mathbf{A}^T\|_\infty \tag{2.17}$$

La norme de Frobenius est matricielle, facile à calculer, mais pas subordonnée :

$$\|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}^H\mathbf{A})} = \sqrt{\sum_{i=i}^{m}\sum_{j=1}^{n} |a_{ij}|^2} \tag{2.18}$$

on voit que la norme de Frobenius est en fait la norme 2 du vecteur composé par les colonnes de $\mathbf{A}$ mises bout-à-bout. Le fait que la norme de Frobenius soit matricielle est simplement la traduction du théorème de Cauchy-Schwarz. Cette norme (et le produit scalaire associé) est fréquemment utilisée en calcul tensoriel (opération de contraction).

Proposition 2.3.4. *On a les encadrements suivants :*

$$\begin{aligned} \|\mathbf{A}\|_2 &\leqslant \|\mathbf{A}\|_F \leqslant \sqrt{n}\|\mathbf{A}\|_2 \\ \frac{1}{\sqrt{n}}\|\mathbf{A}\|_\infty &\leqslant \|\mathbf{A}\|_2 \leqslant \sqrt{m}\|\mathbf{A}\|_\infty \\ \frac{1}{\sqrt{m}}\|\mathbf{A}\|_1 &\leqslant \|\mathbf{A}\|_2 \leqslant \sqrt{n}\|\mathbf{A}\|_1 \end{aligned} \tag{2.19}$$

et les bornes suivantes :

$$\begin{aligned} \|\mathbf{A}\|_2 &\leqslant \sqrt{\|\mathbf{A}\|_1 \|\mathbf{A}\|_\infty} \\ \max_{i,j} |a_{ij}| \leqslant \|\mathbf{A}\|_2 &\leqslant \sqrt{mn} \max_{i,j} |a_{ij}| \end{aligned} \tag{2.20}$$

Proposition 2.3.5.

- *La norme 2 est égale à la plus grande valeur singulière de la matrice* $\|\mathbf{A}\|_2 = \sigma_1 = \sqrt{\rho(\mathbf{A}^H\mathbf{A})} = \sqrt{\rho(\mathbf{A}\mathbf{A}^H)}$,
- *la norme de Frobenius est égale à la racine de la somme des carrés des valeurs singulières* $\|\mathbf{A}\|_F = \sqrt{\sum_i \sigma_i^2}$.
- *Si* $\mathbf{A}$ *est normale on a* $\|\mathbf{A}\|_2 = \rho(\mathbf{A}) = \max(|lambda_i|)$,
- *en particulier si* $\mathbf{A}$ *est unitaire on a* $\|\mathbf{A}\|_2 = 1$.
- *La norme 2 et la norme de Frobenius sont inchangées par multiplication par une matrice unitaire (les transformations unitaires sont des isométries au sens de ces deux normes).*

2.3.3 Approximation du rayon spectral par une norme, suites de matrices

Dans la suite $\mathbf{A}$ est une matrice carrée

Proposition 2.3.6.

- *Pour toute norme matricielle* $\rho(A) \leqslant \|\mathbf{A}\|$.
- *Etant donnée une matrice* $\mathbf{A}$ *et un nombre* $\varepsilon > 0$ *il existe au moins une norme subordonnée telle que* $\|\mathbf{A}\| \leqslant \rho(A) + \varepsilon$.

Démonstration.

Soit $\mathbf{p} \neq 0$ un vecteur tel que $\mathbf{Ap} = \lambda\mathbf{p}$ avec $|\lambda| = \rho(\mathbf{A})$ et soit $\mathbf{q}$ un vecteur non nul alors la matrice $\mathbf{pq}^H$ est non nulle et $\rho(\mathbf{A})\|\mathbf{pq}^H\| = \|\lambda\mathbf{pq}^H\| = \|\mathbf{Apq}^H\| \leqslant \|\mathbf{A}\|\|\mathbf{pq}^H\|$.

Soit $\mathbf{A}$ une matrice et $\mathbf{U}$ une matrice inversible telle que $\mathbf{T} = \mathbf{U}^{-1}\mathbf{AU}$ soit triangulaire supérieure, les valeurs propres apparaissent sur la diagonale. On définit $\mathbf{D}_\delta = \operatorname{diag} 1, \delta, \delta^2, \ldots, \delta^{n-1}$ alors $(\mathbf{UD}_\delta)^{-1}\mathbf{A}(\mathbf{UD}_\delta)$ a la même diagonale que $\mathbf{T}$ et les coefficients extra diagonaux multipliés par une puissance de δ (δ^j où j est le numéro de la surdiagonale), on peut choisir δ tel que $\sum_{j=i+1}^{n} |\delta^{j-1} t_{ij}| \leqslant \epsilon$ pour $1 \leqslant i < n$, et alors $\mathbf{b} \mapsto \|\mathbf{b}\| = \|(\mathbf{UD}_\delta)^{-1}\mathbf{B}(\mathbf{UD}_\delta)\|_\infty$ répond à la question, c'est la norme subordonnée à la norme vectorielle $\mathbf{v} \mapsto \|(\mathbf{UD}_\delta)^{-1}\mathbf{v}\|_\infty$. □

Proposition 2.3.7. *Soit* $\mathbf{B}$ *une matrice carrée. On a les équivalences suivantes*

- $\lim_{k\to\infty} \mathbf{B}^k = 0$,
- $\lim_{k\to\infty} \mathbf{B}^k\mathbf{v} = 0$ *pour tout vecteur* $\mathbf{v}$,
- $\rho(\mathbf{B}) < 1$,
- $\|\mathbf{B}\| < 1$ *pour au moins une norme subordonnée.*

Proposition 2.3.8. *Soit* $\mathbf{B}$ *une matrice carrée, pour toute norme matricielle on a :* $\lim_{k\to\infty} \|\mathbf{B}^k\|^{1/k} = \rho(\mathbf{B})$

Démonstration. On a $\rho(\mathbf{B}) \leqslant \|\mathbf{B}\|$ et comme $\rho(\mathbf{B}^k)^{1/k} = \rho(\mathbf{B})$, on a $\rho(\mathbf{B}) \leqslant \|\mathbf{B}^k\|^{1/k}$. Soit $\mathbf{B}_\epsilon = \frac{\mathbf{B}}{\rho(\mathbf{B})+\epsilon}$, on a $\rho(\mathbf{B}_\epsilon) < 1$ donc $\mathbf{B}_\epsilon^k \to 0$ et donc il existe un entier l tel que pour $k \geqslant n$ $\|\mathbf{B}_\epsilon\| = \frac{\|\mathbf{B}^k\|}{(\rho(\mathbf{B})+\epsilon)^k} \leqslant 1$ ce qui donne $\|\mathbf{B}^k\|^{1/k} \leqslant \rho(\mathbf{B}) + \epsilon$ et qui permet de conclure. □

On cherche fréquemment à inverser des systèmes perturbés $(\mathbf{A} - \delta)$ où $\mathbf{A}$ est supposée d'inverse connue. Le système se réécrit $(\mathbf{A} - \delta)^{-1} = \mathbf{A}^{-1}(\mathbf{I} - \mathbf{A}^{-1}\delta)^{-1}$. D'où l'importance des systèmes de la forme $(\mathbf{I} - \mathbf{B})$.

Proposition 2.3.9. *La norme de* $\mathbf{B}$ *donne une information sur l'inversibilité du système perturbé :*

- *soit une norme subordonnée telle que* $\|\mathbf{B}\| < 1$ *alors* $(\mathbf{I} - \mathbf{B})$ *est inversible, on a le développement de Neumann* $(\mathbf{I} - \mathbf{B})^{-1} = \sum_{k=0}^{\infty} \mathbf{B}^k$ *et* $\|(\mathbf{I} - \mathbf{B})^{-1}\| \leqslant \frac{1}{1-\|\mathbf{B}\|}$,
- *si* $(\mathbf{I} - \mathbf{B})$ *est singulier alors nécessairement* $\|\mathbf{B}\| \geqslant 1$ *pour toute norme matricielle.*

Démonstration.

- $(\mathbf{I}-\mathbf{B})\mathbf{u}=0 \Rightarrow \|u\| = \|\mathbf{B}\mathbf{u}\|$ mais si $\|\mathbf{B}\| < 1$ et $\mathbf{u} \neq 0$ alors $\|\mathbf{B}\mathbf{u}\| \leqslant \|\mathbf{B}\|\|\mathbf{u}\| < \|\mathbf{u}\|$ on a donc nécessairement $\mathbf{u} = 0$ et donc $(\mathbf{I}-\mathbf{B})$ est inversible. On a alors

$$\begin{aligned}(\mathbf{I}-\mathbf{B})^{-1} &= \mathbf{I} + \mathbf{B}(\mathbf{I}-\mathbf{B})^{-1} \\ \|(\mathbf{I}-\mathbf{B})^{-1}\| &\leqslant 1 + \|\mathbf{B}\|\|(\mathbf{I}-\mathbf{B})^{-1}\| \\ \|(\mathbf{I}-\mathbf{B})^{-1}\| &\leqslant \frac{1}{1-\|\mathbf{B}\|}\end{aligned}$$

- $(\mathbf{I}-\mathbf{B})$ singulier revient à dire que 1 est valeur propre de $\mathbf{B}$ donc $\|\mathbf{B}\| \geqslant \rho(\mathbf{B}) \geqslant 1$. □

Proposition 2.3.10 (Développement en série entière)**.** *Soit une série entière $\sum \alpha_k x^k$ de rayon de convergence R, si une matrice carrée $\mathbf{A}$ a un rayon spectral $\rho(\mathbf{A})$ strictement inférieur à R alors la série entière de matrice $\sum \alpha_k A^k$ converge. Cette proposition généralise le résultat sur le développement de Neumann.*

Démonstration. La démonstration s'appuie sur l'écriture de la matrice sous forme de Jordan et le calcul des puissances d'une telle matrice. □

Chapitre 3

Conditionnement

On a vu que les calculs numériques impliquent nécessairement la manipulation de données inexactes (pour cause de propagation d'arrondi ou d'utilisation d'algorithmes itératifs). Un problème important est de quantifier l'influence de ces inexactitudes sur les calculs, celle-ci est caractérisée par le conditionnement des problèmes (en anglais *condition number*, en général noté κ).

On caractérise généralement un algorithme de la façon suivante : soit $y = f(x)$ le résultat de l'évaluation exacte de f en x. Soit $\hat{y}$ le résultat de l'évaluation en précision finie. Soit Δx le nombre tel que l'évaluation exacte de f en $x + \Delta x$ donnerait $\hat{y}$. Voir la figure 3.1

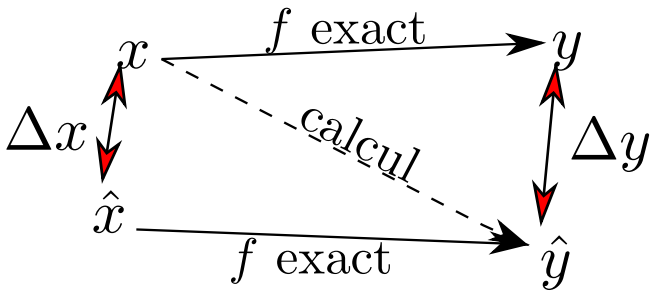


FIGURE 3.1 – Erreurs lors du calcul $y = f(x)$

On appelle $y - \hat{y}$ l'erreur directe (*forward error*) et Δx l'erreur rétrograde (*backward error*). Un algorithme tel que Δx reste faible est dit rétrogradement stable.

Le conditionnement fait le lien entre l'erreur directe et l'erreur rétrograde. En effet, on a :

$$\begin{aligned} \hat{y} - y = f(x + \Delta x) - f(x) &= f'(x)\Delta x + O(\Delta x^2) \\ \frac{\hat{y} - y}{y} &= \frac{x f'(x)}{f(x)} \frac{\Delta x}{x} + O(\Delta x^2) \end{aligned} \tag{3.1}$$

$\kappa = \left| \frac{x f'(x)}{f(x)} \right|$ est le conditionnement de l'algorithme, il donne la variation relative maximale, il borne la propagation des erreurs. On a la règle empirique :

$$\text{erreur directe} \lesssim \kappa \times \text{erreur retrograde} \tag{3.2}$$

On peut donc parler du conditionnement de n'importe quel problème mathématique (calcul des zéros d'une fonction, recherche de minimal, ...). Par la suite on se focalise sur deux problèmes fondamentaux du calcul numérique : la résolution de systèmes linéaires (abusivement nommés inversion de matrice) et le calcul de valeurs et vecteurs propres.

3.1 Conditionnement d'un système linéaire

Exemple 3.1.1. Cet exemple a été proposé par R.S. Wilson. Soit

$$\mathbf{A} = \begin{pmatrix} 10 & 7 & 8 & 7 \\ & 5 & 6 & 5 \\ & & 10 & 9 \\ \text{sym} & & & 10 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix} \quad \delta\mathbf{b} = \begin{pmatrix} 0.1 \\ -0.1 \\ 0.1 \\ -0.1 \end{pmatrix} \quad \mathbf{\Delta A} = \begin{pmatrix} 0 & 0 & 0.1 & 0.2 \\ 0.08 & 0.04 & 0 & 0 \\ 0 & -0.02 & -0.11 & 0 \\ -0,01 & -0,01 & 0 & -0,02 \end{pmatrix} \tag{3.3}$$

On résout $\mathbf{Au} = \mathbf{b}$, $\mathbf{A}(\mathbf{u} + \delta\mathbf{u}) = \mathbf{b} + \delta\mathbf{b}$ et $(\mathbf{A} + \mathbf{\Delta A})(\mathbf{u} + \mathbf{\Delta u}) = \mathbf{b}$. On a

$$\mathbf{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \delta\mathbf{u} = \begin{pmatrix} 8.2 \\ -13.6 \\ 3.5 \\ -2.1 \end{pmatrix}, \quad \mathbf{\Delta u} = \begin{pmatrix} -82 \\ 136 \\ -35 \\ 21 \end{pmatrix}, \quad \text{au passage } \mathbf{A}^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ & 68 & -17 & 10 \\ & & 5 & -3 \\ \text{sym} & & & 2 \end{pmatrix} \tag{3.4}$$

On voit qu'une perturbation de 3.10^{-3} sur la norme 2 du second membre conduit à une variation de 8 sur la norme 2 de la solution et qu'une perturbation de 7.10^{-3} sur la norme 2 de la matrice conduit à une variation de 160 sur la norme 2 de la solution (ce qui colle assez bien avec les propriétés du conditionnement décrites dans la prochaine section, $\kappa_2(\mathbf{A}) \simeq 3000$ et $\kappa_1(\mathbf{A}) = \kappa_\infty(\mathbf{A}) \simeq 4500$).

3.1.1 Caractérisation classique

Définition 3.1.1. Pour une norme subordonnée, on appelle conditionnement de la matrice $\mathbf{A}$ le nombre $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$, remarquons dès à présent que le conditionnement est une notion dépendante de la norme utilisée (si besoin est, on peut le noter $\kappa_p(\mathbf{A})$). Par convention une matrice singulière a un conditionnement infini.

Soit une matrice $\mathbf{A}$ *inversible*, on compare la solution d'un système linéaire dont on a perturbé le second membre avec le système original.

$$\begin{aligned} \mathbf{Au} &= \mathbf{b} \\ \mathbf{A}(\mathbf{u} + \delta\mathbf{u}) &= \mathbf{b} + \delta\mathbf{b} \end{aligned} \tag{3.5}$$

Proposition 3.1.1. *On suppose $\mathbf{b} \neq 0$, pour n'importe quelle norme subordonnée, on a :*

$$\frac{\|\delta\mathbf{u}\|}{\|\mathbf{u}\|} \leqslant \kappa(\mathbf{A}) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|} \tag{3.6}$$

c'est une borne exacte (il existe une perturbation pour laquelle le résultat est une égalité).

Démonstration. On a $\delta\mathbf{u} = \mathbf{A}^{-1}\delta\mathbf{b}$ et donc $\|\delta\mathbf{u}\| \leqslant \|\mathbf{A}^{-1}\|\|\delta\mathbf{b}\|$. On a de plus $\|\mathbf{b}\| \leqslant \|\mathbf{A}\|\|\mathbf{u}\|$. □

On étudie maintenant l'effet d'une perturbation de la matrice sur la solution :

$$\begin{aligned} \mathbf{Au} &= \mathbf{b} \\ (\mathbf{A} + \mathbf{\Delta A})(\mathbf{u} + \mathbf{\Delta u}) &= \mathbf{b} \end{aligned} \tag{3.7}$$

Proposition 3.1.2. *En supposant $\mathbf{b} \neq 0$, on a :*

$$\frac{\|\mathbf{\Delta u}\|}{\|\mathbf{u} + \mathbf{\Delta u}\|} \leqslant \kappa(\mathbf{A}) \frac{\|\mathbf{\Delta A}\|}{\|\mathbf{A}\|} \tag{3.8}$$

là encore l'égalité peut être atteinte.

Démonstration. Le système donne $\mathbf{A\Delta u} + \mathbf{\Delta A}(\mathbf{u} + \mathbf{\Delta u}) = 0$ et donc $\mathbf{\Delta u} = -\mathbf{A}^{-1}\mathbf{\Delta A}(\mathbf{u} + \mathbf{\Delta u})$ qui conduit à $\|\mathbf{\Delta u}\| \leqslant \|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\|\|\mathbf{u} + \mathbf{\Delta u}\| = \kappa(\mathbf{A}) \frac{\|\mathbf{\Delta A}\|}{\|\mathbf{A}\|} \|\mathbf{u} + \mathbf{\Delta u}\|$ □

Proposition 3.1.3. *On peut arranger le premier membre et obtenir :*

$$\frac{\|\mathbf{\Delta u}\|}{\|\mathbf{u}\|} \leqslant \kappa(\mathbf{A}) \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} (1 + O(\|\mathbf{\Delta A}\|) \tag{3.9}$$

Démonstration. On suppose que la perturbation reste petite telle que $\|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\| < 1$ de sorte que $(\mathbf{I} + \mathbf{A}^{-1}\mathbf{\Delta A})$ est inversible et $\|(\mathbf{I} + \mathbf{A}^{-1}\mathbf{\Delta A})^{-1}\| \leqslant \frac{1}{1-\|\mathbf{A}^{-1}\mathbf{\Delta A}\|} \leqslant \frac{1}{1-\|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\|}$. On a $\mathbf{\Delta u} = -\mathbf{A}^{-1}\mathbf{\Delta A}(\mathbf{u} + \mathbf{\Delta u})$ et par ailleurs $(\mathbf{A} + \mathbf{\Delta A})(\mathbf{u} + \mathbf{\Delta u}) = \mathbf{Au}$ donc $\mathbf{u} + \mathbf{\Delta u} = (\mathbf{I} + \mathbf{A}^{-1}\mathbf{\Delta A})^{-1}\mathbf{u}$. Donc

$$\begin{aligned}
\mathbf{\Delta u} &= -\mathbf{A}^{-1}\mathbf{\Delta A}(\mathbf{I} + \mathbf{A}^{-1}\mathbf{\Delta A})^{-1}\mathbf{u} \\
\|\mathbf{\Delta u}\| &\leqslant \frac{\|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\|}{1 - \|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\|}\|\mathbf{u}\| \\
\frac{\|\mathbf{\Delta u}\|}{\|\mathbf{u}\|} &\leqslant \kappa(\mathbf{A})\frac{\|\mathbf{\Delta A}\|}{\|\mathbf{A}\|}\left(\frac{1}{1 - \|\mathbf{A}^{-1}\|\|\mathbf{\Delta A}\|}\right)
\end{aligned}$$

□

Proposition 3.1.4. *On a les résultats importants suivants* $\forall \mathbf{A}$ *:*

$$\begin{aligned}
\kappa(\mathbf{A}) &\geqslant 1 \\
\kappa(\mathbf{A}) &= \kappa(\mathbf{A^{-1}}) \\
\kappa(\alpha\mathbf{A}) &= \kappa(\mathbf{A}),\ \textit{pour tout scalaire } \alpha \\
\kappa_2(\mathbf{A}) &= \frac{\sigma_1}{\sigma_n},\quad (\sigma_1 \leqslant \cdots \leqslant \sigma_n)\ \textit{valeurs singulières de } \mathbf{A} \\
\textit{Si } \mathbf{A} \textit{ est normale, } \kappa_2(\mathbf{A}) &= \frac{\max|\lambda_i|}{\min|\lambda_i|},\quad (\lambda_i)\ \textit{valeurs propres de } \mathbf{A} \\
\textit{Si } \mathbf{A} \textit{ est unitaire ou orthogonale, } \kappa_2(\mathbf{A}) &= 1
\end{aligned} \tag{3.10}$$

Proposition 3.1.5. $\kappa_2(\mathbf{A})$ *est invariant par transformation unitaire :*

$$\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I} \Rightarrow \kappa_2(\mathbf{A}) = \kappa_2(\mathbf{UA}) = \kappa_2(\mathbf{AU}) = \kappa_2(\mathbf{U^H A U}) \tag{3.11}$$

Ce dernier résultat indique l'importance des transformations unitaires en analyse numérique puisqu'elles ne détériorent pas le conditionnement (voir donc les algorithmes faisant intervenir les rotations de Givens ou les réflexions de Householder).

En mécanique des structures il est possible de relier le conditionnement, entre autres, à l'élancement (*aspect ratio*) des structures, à la présence d'hétérogénéités, à l'incompressibilité, aux interfaces irrégulières...

3.1.2 Scaling

La précision de nombreuses opérations dépend du conditionnement de la matrice d'entrée. Par exemple, la factorisation de Cholesky d'une matrice symétrique définie positive $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ est rétrogradement stable (ce qui est parfait pour résoudre un système) mais l'erreur directe est de la forme $\|\mathbf{L} - \hat{\mathbf{L}}\|/\|\mathbf{L}\| \simeq u\kappa(\mathbf{A})$, où u est l'erreur d'arrondi introduite au chapitre précédent. On a donc intérêt à essayer d'améliorer le conditionnement des matrices avant d'appliquer dessus de nombreuses opérations.

Pour cela on utilise des matrices d'équilibrage (*scaling*). On écrit $\mathbf{A} = \mathbf{D_g}\tilde{\mathbf{A}}\mathbf{D_d}$ avec $\mathbf{D_g}$ et $\mathbf{D_d}$ deux matrices diagonales (donc très faciles à manipuler) rendant $\tilde{\mathbf{A}}$ de conditionnement faible. Il n'y a pas de recette absolue, trouver les matrices d'équilibrage qui minimisent le conditionnement est un problème très complexe (qui dépend de la matrice et de la norme utilisée).

Une idée classique consiste à essayer de rendre chaque ligne ou chaque colonne du même ordre de grandeur (en norme infinie). Pour les matrices symétriques définies positives avec de gros coefficient diagonaux on peut prendre la racine de la diagonale.

3.1.3 Autres caractérisations du conditionnement d'une matrice

$$\begin{aligned}
\frac{1}{\kappa_p(\mathbf{A})} &= \min_{\mathbf{A}+\mathbf{E}\ \text{singulière}} \frac{\|\mathbf{E}\|_p}{\|\mathbf{A}\|_p} \\
\kappa(\mathbf{A}) &= \lim_{\delta\to 0}\sup_{\|\mathbf{E}\|\leqslant\delta\|\mathbf{A}\|} \frac{\|(\mathbf{A}+\mathbf{E})^{-1} - \mathbf{A}^{-1}\|}{\delta}\frac{1}{\|\mathbf{A}^{-1}\|}
\end{aligned} \tag{3.12}$$

On retrouve la derivée de Fréchet normalisée de la fonction inverse : $\mathbf{A} \to \mathbf{A}^{-1}$.

3.1.4 Remarque sur le déterminant

Le déterminant n'est pas utilisable pour caractériser la non-inversibilité d'une matrice.

Exemple 3.1.2.

$$\begin{pmatrix} 1 & -1 & \cdots & -1 \\ 0 & 1 & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathbb{R}^{n\times n} \tag{3.13}$$

a un déterminant de 1 et un conditionnement de $n2^{n-1}$.

Exemple 3.1.3. $\mathbf{D}_n = \mathrm{diag}(10^{-1}, \cdots, 10^{-1}) \in \mathbb{R}^{n\times n}$ a un $\kappa_p(\mathbf{D_n}) = 1$ et un déterminant $\det(\mathbf{D}_n) = 10^{-n}$.

3.2 Conditionnement d'un problème aux valeurs propres

Exemple 3.2.1.

$$\begin{pmatrix} 0 & & & & \varepsilon \\ 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{pmatrix} \tag{3.14}$$

Si $\varepsilon = 0$ toutes les valeurs propres sont nulles, si $\varepsilon = 10^{-n}$ toutes les valeurs propres valent 10^{-1}, soit un conditionnement de 10^{n-1}. De plus si 10^{-n} est hors précision machine (disons $n \simeq 40$), alors il sera arrondi à 0 ce qui signifie que les valeurs propres seront fausses de 10^{-1} (erreur absolue).

Soit $\mathbf{A}$ une matrice diagonalisable, $\mathbf{P}$ une matrice telle que

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathrm{diag}\,\lambda_i = \mathbf{D} \tag{3.15}$$

et une norme matricielle telle que

$$\|\mathrm{diag}(d_i)\| = \max_i |d_i| \tag{3.16}$$

pour toute matrice diagonale (propriété vérifiée par les normes 1, 2 et ∞).

Proposition 3.2.1. *Alors pour toute matrice $\delta\mathbf{A}$,*

$$\begin{gathered} \mathrm{sp}(\mathbf{A}+\delta\mathbf{A}) \subset \bigcup_{i=1}^{n} D_i \\ \text{avec } D_i = \{z \in \mathbb{C};\ |z-\lambda_i| \leqslant \kappa(\mathbf{P})\|\delta\mathbf{A}\|\} \end{gathered} \tag{3.17}$$

Démonstration. Soit λ une valeur propre de $(\mathbf{A}+\delta\mathbf{A})$ avec $\lambda \neq \lambda_i$ quel que soit i (la valeur propre a donc été perturbée par $\delta\mathbf{A}$). On a donc $(\mathbf{D}-\lambda\mathbf{I})$ inversible et

$$\mathbf{P}^{-1}(\mathbf{A}+\delta\mathbf{A}-\lambda\mathbf{I})\mathbf{P} = \mathbf{D}-\lambda\mathbf{I}+\mathbf{P}^{-1}\delta\mathbf{A}\mathbf{P} = (\mathbf{D}-\lambda\mathbf{I})(\mathbf{I}+(\mathbf{D}-\lambda\mathbf{I})^{-1}\mathbf{P}^{-1}\delta\mathbf{A}\mathbf{P})$$

La matrice $\mathbf{I}+(\mathbf{D}-\lambda\mathbf{I})^{-1}\mathbf{P}^{-1}\delta\mathbf{A}\mathbf{P}$ n'est pas inversible (puisque $(\mathbf{A}+\delta\mathbf{A}-\lambda\mathbf{I})$ ne l'est pas) et donc

$$1 \leqslant \|(\mathbf{D}-\lambda\mathbf{I})^{-1}\mathbf{P}^{-1}\delta\mathbf{A}\mathbf{P}\| \leqslant \|(\mathbf{D}-\lambda\mathbf{I})^{-1}\|\|\mathbf{P}^{-1}\|\|\delta\mathbf{A}\|\|\mathbf{P}\| \tag{3.18}$$

or d'après l'hypothèse sur la norme matricielle$\|(\mathbf{D}-\lambda\mathbf{I})^{-1}\| = \frac{1}{\min|\lambda_i-\lambda|}$ donc il existe au moins un i tel que

$$|\lambda_i-\lambda| \leqslant \|\mathbf{P}^{-1}\|\|\delta\mathbf{A}\|\|\mathbf{P}\| = \kappa(\mathbf{P})\|\delta\mathbf{A}\| \tag{3.19}$$

□

Donc le conditionnement d'un problème aux valeurs propres est la valeur $\inf\{\kappa(\mathbf{P});\ \mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathrm{diag}(\lambda_i)\}$. En conséquence une matrice normale (et en particulier hermitienne) est parfaitement conditionnée (en norme 2) pour un problème aux valeurs propres puisqu'elle diagonalise dans un changement de base unitaire.

Proposition 3.2.2. *Si* $\mathbf{A}$ *et* $\mathbf{B} = \mathbf{A} + \delta\mathbf{A}$ *sont deux matrices hermitiennes, de valeurs propres (ordonnées) respectives* $(\alpha)_i$ *et* $(\beta)_i$

$$|\alpha_k - \beta_k| \leqslant \|\delta\mathbf{A}\|_2 \tag{3.20}$$

Ce résultat est plus fort que le précédent qui dirait simplement que pour tout k *il existe* i *tel que* $|\alpha_k - \beta_i| \leqslant \|\delta\mathbf{A}\|_2$.

Démonstration. On utilise le quotient de Rayleigh, E_k est l'espace propre engendré par les k premiers vecteurs propres et $\mathcal{E}_k$ l'ensemble des sous-espaces de dimension k, on a

$$\begin{aligned}
\beta_k = \min_{\mathbb{V}\in\mathcal{E}_{k-1}} \max_{\mathbf{v}\perp\mathbb{V}} R_{\mathbf{B}}(\mathbf{v}) &\leqslant \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} R_{\mathbf{B}}(\mathbf{v}) = \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} \left(R_{\mathbf{A}}(\mathbf{v}) + R_{\delta\mathbf{A}}(\mathbf{v})\right) \\
&\leqslant \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} R_{\mathbf{A}}(\mathbf{v}) + \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} R_{\delta\mathbf{A}}(\mathbf{v}) = \alpha_k + \max_{\mathbf{v}\perp\mathbb{E}_{k-1}} R_{\delta\mathbf{A}}(\mathbf{v}) \\
&\leqslant \alpha_k + \max_{\mathbf{v}\in\mathbb{E}} R_{\delta\mathbf{A}}(\mathbf{v}) \leqslant \alpha_k + \|\delta\mathbf{A}\|_2
\end{aligned}$$

On peut faire le même calcul en échangeant β_k et α_k (en utilisant $-\delta\mathbf{A}$), ce qui permet de conclure. □

Chapitre 4

Généralités sur la résolution de systèmes linéaires

Il est fondamental de noter que résoudre (on dit aussi, à tort, « inverser ») un système n'implique pas de calculer la matrice inverse. Calculer une inverse revient à résoudre n systèmes (un par vecteur de base) et est donc une opération très coûteuse et souvent inutile.

Les résultats donnés par la suite dans $\mathbb{C}$ restent valables dans $\mathbb{R}$ (toutes les matrices impliquées sont alors réelles).

On veut résoudre $\mathbf{Ax} = \mathbf{b}$. On introduit des bases des noyaux à droite et à gauche $\mathbf{AN}_d = 0$ et $\mathbf{A}^H\mathbf{N}_g = 0$. On rappelle l'alternative de Fredholm :

— $\mathbf{b} \in \mathrm{Im}(\mathbf{A}) \Leftrightarrow \mathbf{N}_g^H\mathbf{b} = 0$ alors le problème a une solution définie à un élément de $\mathrm{Ker}_d(\mathbf{A})$ près (que l'on peut noter $\mathbf{N}_d\boldsymbol{\alpha}$ avec $\boldsymbol{\alpha}$ quelconque). Si $\mathrm{Ker}_d(\mathbf{A}) \neq \{0\}$, on parle de problème sous-déterminé.

— $\mathbf{b} \notin \mathrm{Im}(\mathbf{A}) \Leftrightarrow \mathbf{N}_g^H\mathbf{b} \neq 0$ alors le problème n'a pas de solution, on parle de problème surdéterminé.

Remarquons qu'il arrive que le second membre ne soit pas entièrement connu et que cette partie inconnue serve à rendre le problème bien posé : on résout en fait $\mathbf{Ax} = \mathbf{b} + \mathbf{y}$ où $\mathbf{y}$ est inconnu et doit satisfaire $\mathbf{N}_g^H(\mathbf{b} + \mathbf{y}) = 0$; la détermination de $\mathbf{y}$ est un problème sous-déterminé dont on parlera plus tard.

Dans le cas des systèmes rectangulaires, on sait qu'il y a nécessairement des noyaux à gauche ou à droite et que soit il n'y a pas de solution (en général) soit la solution n'est pas unique.

On verra que la résolution de systèmes rectangulaires implique (au moins formellement) la résolution de systèmes carrés $\mathbf{A}^H\mathbf{CA}$, où $\mathbf{C}$ est une matrice symétrique définie positive, on a le résultat fondamental suivant :

Proposition 4.0.1. *Pour toute matrice* $\mathbf{A}$*,* $\mathbf{C}$ *étant une matrice symétrique (hermitienne) définie positive, on a* $\mathrm{Ker}(\mathbf{A^HCA}) = \mathrm{Ker}(\mathbf{A})$ *et* $\mathrm{Im}(\mathbf{A^HCA}) = \mathrm{Im}(\mathbf{A^H})$*.*

Démonstration. On a bien sûr $\mathrm{Ker}(\mathbf{A}) \subset \mathrm{Ker}(\mathbf{A^HCA})$; si on prend un vecteur $\mathbf{x}$ de $\mathrm{Ker}(\mathbf{A^HCA})$ alors $0 = \mathbf{x}^H\mathbf{A}^H\mathbf{CAx} = \|\mathbf{Ax}\|_{\mathbf{C}} \Rightarrow \mathbf{x} \in \mathrm{Ker}(\mathbf{A})$.

On a bien sûr $\mathrm{Im}(\mathbf{A^HCA}) \subset \mathrm{Im}(\mathbf{A^H})$; si $\mathbf{y} \in \mathrm{Im}(\mathbf{A^H})$ alors $\exists \mathbf{x}/\mathbf{A}^H\mathbf{x} = \mathbf{y}$. D'après la supplémentarité image/noyau, il existe des uniques $(\mathbf{x}_1, \mathbf{x}_2)$ tels que $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$ avec $\mathbf{x}_1 \in \mathrm{Im}(\mathbf{CA})$, $\mathbf{x}_1 = \mathbf{CAz}$ et $\mathbf{x}_2 \in \mathrm{Ker}(\mathbf{A^HC})$, on a donc $\mathbf{y} = \mathbf{A}^H\mathbf{CAz}$ et l'inclusion est réciproque. □

Définition 4.0.1. On dira d'une matrice qu'elle est de rang plein si son rang est égal à la plus petite de ses dimensions. Dans ce cas un des noyaux (à gauche ou à droite suivant la forme de la matrice) est réduit à 0. En anglais suivant la forme de la matrice, on parlera de *full-column-ranked matrix* ou *full-row-ranked matrix*.

4.1 Problème surdéterminé

Si $\mathbf{b} \notin \mathrm{Im}(\mathbf{A})$, ce qui implique $\mathrm{Ker}_g(\mathbf{A}) \neq \{0\}$ et ce qui est typiquement le cas si la matrice est rectangulaire $m \times n$ avec plus de lignes (m) que de colonnes ($n < m$, pas assez d'inconnues, trop d'équations), le problème peut être compris dans un sens étendu, et on cherche en fait à minimiser un résidu : $\mathbf{x} = \arg\min_{\mathbf{y}} \|\mathbf{b} - \mathbf{Ay}\|$.

Le résultat dépend du choix de la norme.

$$\operatorname{argmin} \left\| \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} x - \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \right\|_p \qquad \text{avec } b_1 \geqslant b_2 \geqslant b_3 \qquad \begin{cases} p = 1 & x = b_2 \\ p = 2 & x = (b_1 + b_2 + b_3)/3 \\ p = \infty & x = (b_1 + b_3)/2 \end{cases} \tag{4.1}$$

Les normes 1 et ∞ étant non différentiables elles sont plus difficiles à manipuler. On préfère donc en général la norme 2 qui conduit aux problèmes de moindres carrés, ou une norme associée à une matrice $\mathbf{C}$ symétrique (hermitienne) définie positive.

La différentiation de l'expression à minimiser conduit à :

$$\begin{aligned} 0 = d\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_C^2 &= d\left((\mathbf{b} - \mathbf{A}\mathbf{x})^T \mathbf{C}(\mathbf{b} - \mathbf{A}\mathbf{x})\right) \\ &= d\left((\mathbf{b}^T\mathbf{C}\mathbf{b} + \mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{A}^T\mathbf{C}\mathbf{b})\right) = 2(d\mathbf{x})^T\left(\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} - \mathbf{A}^T\mathbf{C}\mathbf{b}\right) \end{aligned} \tag{4.2}$$

Donc le problème de minimisation se ramène à résoudre $\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{C}\mathbf{b}$ qui est un problème carré bien posé dans le sens où le second membre est dans l'image de l'opérateur. Quand $\mathbf{C} = \mathbf{I}$ on parle d'équation normale. La matrice $\mathbf{A}^T\mathbf{C}\mathbf{A}$ est symétrique positive, quand $\mathbf{A}$ est de rang plein elle est définie et se prête bien à une factorisation de Choleski. Le problème d'une telle approche est le mauvais conditionnement puisque $\kappa_2(\mathbf{A}^\mathbf{T}\mathbf{A}) = \kappa_2(\mathbf{A})^2$.

4.2 Problème sous-déterminé

Quitte à étudier l'équation normale d'un problème surdéterminé, on se place dans le cas où $\mathbf{b} \in \mathrm{Im}(\mathbf{A})$ mais $\dim(\mathrm{Ker}_d(\mathbf{A})) > 0$, le problème est dit sous-déterminé, il existe une infinité de solution. C'est typiquement le cas des matrices avec plus de colonne que de lignes (pas assez d'équations).

Pour rendre le problème à solution unique, on peut chercher à minimiser la norme du vecteur solution : $\mathbf{x} = \arg\min_{\mathbf{A}\mathbf{y}=\mathbf{b}} \|\mathbf{y}\|$. La minimisation se faisant sur le noyau à droite qui est un sous-espace vectoriel et donc un ensemble convexe, on sait qu'il existe un unique élément de norme minimale. En général on choisit une norme définie par une matrice $\mathbf{C}$ symétrique (hermitienne) définie positive. On introduit un Lagrangien pour réaliser la minimisation sous contrainte :

$$\mathcal{L}(\mathbf{y}, \boldsymbol{\mu}) = \|\mathbf{y}\|_\mathbf{C}^2 + (\mathbf{A}\mathbf{y} - \mathbf{b})^T \boldsymbol{\mu} \tag{4.3}$$

dont la stationnarité est réalisée $(\mathbf{x}, \boldsymbol{\lambda})$ donnés par :

$$\left(\mathbf{A}\mathbf{C}^{-1}\mathbf{A}^T\right)\boldsymbol{\lambda} = -\mathbf{b} \qquad \text{et} \qquad \mathbf{x} = -\mathbf{C}^{-1}\mathbf{A}^T\boldsymbol{\lambda} \tag{4.4}$$

D'après l'hypothèse de départ $\mathbf{b} \in \mathrm{Im}(\mathbf{A})$, le problème carré de matrice symétrique positive $\left(\mathbf{A}\mathbf{C}^{-1}\mathbf{A}^T\right)$ est bien posé, le problème est défini si $\mathbf{A}$ est de rang plein. Notons que si la matrice n'est pas de rang plein $\boldsymbol{\lambda}$ n'est pas défini de manière unique mais $\mathbf{x}$ l'est.

4.3 Élimination par blocs – complément de Schur

Soit le découpage par blocs suivant :

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \tag{4.5}$$

où $\mathbf{A}_{11}$ est supposée inversible, on définit alors le complément de Schur de $\mathbf{A}_{11}$ dans $\mathbf{A}$ que l'on note $\mathbf{A}/\mathbf{A}_{11}$

$$\mathbf{A}/\mathbf{A}_{11} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \tag{4.6}$$

Proposition 4.3.1 (Formule de diagonalisation par blocs de Aitken)**.**

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{I} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}/\mathbf{A}_{11} \end{pmatrix} \tag{4.7}$$

Corollaire 4.3.2.

- $\det(\mathbf{A}) = \det(\mathbf{A_{11}}) \det(\mathbf{A}/\mathbf{A_{11}})$

- $\mathrm{rg}(\mathbf{A}) = \mathrm{rg}(\mathbf{A_{11}}) + \mathrm{rg}(\mathbf{A}/\mathbf{A_{11}})$
- $\mathrm{rg}(\mathbf{A}) = \mathrm{rg}(\mathbf{A_{11}}) \quad \Rightarrow \quad \mathbf{A}/\mathbf{A_{11}} = 0$
- $\dim(\mathrm{Ker}(\mathbf{A})) = \dim(\mathrm{Ker}(\mathbf{A}/\mathbf{A_{11}}))$

Si $\mathbf{A}/\mathbf{A_{11}}$ est inversible on a également :

$$\begin{aligned}
\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} &= \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}/\mathbf{A}_{11} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \\
\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} &= \begin{pmatrix} \mathbf{I} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{A}_{11}^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{A}/\mathbf{A}_{11})^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{I} \end{pmatrix} \\
&= \begin{pmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}/\mathbf{A}_{11})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}/\mathbf{A}_{11})^{-1} \\ -(\mathbf{A}/\mathbf{A}_{11})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & (\mathbf{A}/\mathbf{A}_{11})^{-1} \end{pmatrix}
\end{aligned} \tag{4.8}$$

Proposition 4.3.3.

$$(\mathbf{A}/\mathbf{A}_{11})^{-1} = (\mathbf{A}^{-1})_{22} \tag{4.9}$$

Si on suppose de plus que $\mathbf{A}_{22}$ est inversible, on peut calculer le complément de Schur $\mathbf{A}/\mathbf{A}_{22}$ et on a

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} (\mathbf{A}/\mathbf{A}_{22})^{-1} & -(\mathbf{A}/\mathbf{A}_{22})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}(\mathbf{A}/\mathbf{A}_{22})^{-1} & \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{21}(\mathbf{A}/\mathbf{A}_{22})^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \end{pmatrix} \tag{4.10}$$

En égalant les deux expressions, on obtient notamment :

$$(\mathbf{A}/\mathbf{A}_{22})^{-1} = (\mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1} = \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}(\mathbf{A}/\mathbf{A}_{11})^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} \tag{4.11}$$

qui est exactement la formule de Woodbury (Sherman, Morrisson, Duncan, Banachiewicz).

Proposition 4.3.4 (Formule du quotient)**.** *Soit*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \mathbf{A}_{13} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} \\ \mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{A}}_{11} & \tilde{\mathbf{A}}_{13} \\ \tilde{\mathbf{A}}_{31} & \mathbf{A}_{33} \end{pmatrix} \text{ avec } \tilde{\mathbf{A}}_{11} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix},\ \tilde{\mathbf{A}}_{13} = \begin{pmatrix} \mathbf{A}_{13} \\ \mathbf{A}_{23} \end{pmatrix} \tag{4.12}$$

alors

$$\mathbf{A}/\tilde{\mathbf{A}}_{11} = (\mathbf{A}/\mathbf{A}_{11})/(\tilde{\mathbf{A}}_{11}/\mathbf{A}_{11}) \tag{4.13}$$

Cette formule est fondamentale dans la compression de l'enchaînement des calculs dans une élimination de Gauss : éliminer une colonne puis une autre est équivalente à éliminer deux colonnes d'un coup, d'où la possibilité de calculer un complément de Schur en bloquant un processus de Gauss (après numérotation convenable des degrés de libertés). Elle permet aussi de comprendre les méthodes de « recondensation » dans les décompositions de domaine (FETI-DP, BDDC).

Profitons de l'occasion pour rajouter deux propriétés sur les compléments de Schur

Proposition 4.3.5 (Héritage de propriétés)**.** *Le complément de Schur $\mathbf{A}/\mathbf{A}_{11}$ hérite des propriétés suivantes de $\mathbf{A}$:*

- *Symétrie, anti-symétrie, caractère hermitien, caractère anti-hermitien,*
- *Profil diagonal, triangulaire, Hessenberg, tridiagonal*
- *Caractère défini positif,*
- *Dominance de la diagonale (par ligne et/ou par colonne), H-matrices (matrices rendues à diagonale dominante par le produit à gauche ou à droite par une matrice diagonale)*
- *M-matrices (matrices dont les coefficients diagonaux sont strictement positifs et les autres coefficients négatifs ou nuls)*

Proposition 4.3.6 (Formule de l'inertie de Haynsworth)**.** *Pour une matrice hermitienne $\mathbf{A}$, on appelle inertie le triplet de naturels $\mathrm{In}(\mathbf{A}) = (p, z, n)$ donnant le nombre de valeurs propres positives (p), égales à zéro (z) et négatives (n).*

$$\mathrm{In}(\mathbf{A}) = \mathrm{In}(\mathbf{A}_{11}) + \mathrm{In}(\mathbf{A}/\mathbf{A}_{11}) \tag{4.14}$$

4.4 Notion de pseudo-inverse

Pour une matrice $\mathbf{A}$ donnée (rectangulaire ou carrée), une pseudo-inverse (ou inverse généralisée) est une matrice $\mathbf{A}^+$ qui vérifie :

$$\forall \mathbf{y} \in \mathrm{Im}(\mathbf{A}), \mathbf{A}\mathbf{A}^+\mathbf{y} = \mathbf{y} \tag{4.15}$$

Bien sûr si $\mathbf{A}$ est inversible, la pseudo-inverse coïncide avec l'inverse. Sinon, la pseudo-inverse n'est pas unique. Si la matrice $\mathbf{A}$ est de rang r alors il existe une matrice $\mathbf{A}_{11}$ $r \times r$ inversible et deux changements de bases $\mathbf{Q}$ et $\mathbf{P}$ tels que

$$\begin{aligned} \mathbf{A} &= \mathbf{P}\begin{pmatrix} \mathbf{A}_{11} & 0 \\ 0 & 0 \end{pmatrix}\mathbf{Q} \\ \mathbf{A}^+ &= \mathbf{Q}^{-1}\begin{pmatrix} \mathbf{A}_{11}^{-1} & \mathbf{X} \\ \mathbf{Y} & \mathbf{Z} \end{pmatrix}\mathbf{P}^{-1} \text{ où } \mathbf{X}, \mathbf{Y}, \mathbf{Z} \text{ sont quelconques} \end{aligned} \tag{4.16}$$

Proposition 4.4.1.

$$\begin{aligned} \mathbf{A} &= \mathbf{A}(\mathbf{A}^H\mathbf{A})^+(\mathbf{A}^H\mathbf{A}) \\ \mathbf{A}^H &= (\mathbf{A}^H\mathbf{A})(\mathbf{A}^H\mathbf{A})^+\mathbf{A}^H \end{aligned} \tag{4.17}$$

Ces formules montrent que la pseudo-inversion peut se ramener à un problème carré. En particulier, quand $\mathbf{A}$ *est de rang plein alors les matrices carrées à pseudo-inverser sont inversibles au sens usuel.*

Démonstration. On a $\forall \alpha$, $(\mathbf{A}^H\mathbf{A})\alpha = (\mathbf{A}^H\mathbf{A})(\mathbf{A}^H\mathbf{A})^+(\mathbf{A}^H\mathbf{A})\alpha$ donc $\mathbf{A}^H(\mathbf{A} - \mathbf{A}(\mathbf{A}^H\mathbf{A})^+(\mathbf{A}^H\mathbf{A}))\alpha = 0$. Comme $\mathrm{Im}(\mathbf{A}) \perp \mathrm{Ker}(\mathbf{A^H})$ on a nécessairement $\mathbf{A} - \mathbf{A}(\mathbf{A}^H\mathbf{A})^+(\mathbf{A}^H\mathbf{A}) = 0$ □

Les pseudo-inverses peuvent vérifier certaines propriétés additionnelles, la pseudo-inverse la plus riche (et la moins calculable) est celle de Moore-Penrose $\mathbf{A}^+_{MP}$, elle est la seule à vérifier :

$$\begin{aligned} \mathbf{A}\mathbf{A}^+_{MP}\mathbf{A} &= \mathbf{A} & (\mathbf{A}\mathbf{A}^+_{MP})^H &= \mathbf{A}\mathbf{A}^+_{MP} \\ \mathbf{A}^+_{MP}\mathbf{A}\mathbf{A}^+_{MP} &= \mathbf{A}^+_{MP} & (\mathbf{A}^+_{MP}\mathbf{A})^H &= \mathbf{A}^+_{MP}\mathbf{A} \end{aligned} \tag{4.18}$$

La pseudo inverse de Moore-Penrose peut se calculer via une SVD (qui est un cas particulier de changement de base) :

$$\begin{aligned} \mathbf{M} &= \mathbf{U}\begin{pmatrix} \mathbf{\Sigma} & 0 \\ 0 & 0 \end{pmatrix}\mathbf{V}^H \\ \mathbf{M}^+_{MP} &= \mathbf{V}\begin{pmatrix} \mathbf{\Sigma}^{-1} & 0 \\ 0 & 0 \end{pmatrix}\mathbf{U}^H \end{aligned} \tag{4.19}$$

Comme $\mathbf{U}$ et $\mathbf{V}$ sont unitaires, elles ne modifient pas les normes euclidiennes. On en déduit donc que la pseudo-inverse de Moore-Penrose donne la solution de norme minimale du problème aux moindres carrés associé à $\mathbf{A}$.

Un cas particulier important est celui des matrices unitaires (orthogonales) puisque l'on a directement $\mathbf{U}^+_{MP} = \mathbf{U}^H$.

Cela implique pour une factorisation $\mathbf{A} = \mathbf{QR}$, $\mathbf{A}^+_{MP} = \mathbf{R}^+_{MP}\mathbf{Q}^H$.

4.4.1 Formules de calcul

Une façon de trouver une pseudo-inverse s'appuie sur la connaissance des noyaux pour définir un problème régularisé.

Proposition 4.4.2. *On a pour une matrice* $\mathbf{C}$ *quelconque*

$$\begin{pmatrix} \mathbf{A} & \mathbf{N}_g \\ \mathbf{N}_d^H & \mathbf{C} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^+ & ? \\ ? & ? \end{pmatrix} \tag{4.20}$$

Proposition 4.4.3. *Si de plus* $\mathbf{C}$ *est inversible, on peut calculer le complément de Schur (en posant* $\mathbf{C} \leftarrow -\mathbf{C}^{-1}$ *dans la formule précédente), on obtient alors :*

$$\left(\mathbf{A} + \mathbf{N}_g\mathbf{C}\mathbf{N}_d^H\right)^{-1} = \mathbf{A}^+ \tag{4.21}$$

La pseudo-inverse de Moore-Penrose est un cas particulier de la formule précédente (pour $\mathbf{C} = 0$) :

Proposition 4.4.4. *On a*

$$\begin{pmatrix} \mathbf{A} & \mathbf{N}_g \\ \mathbf{N}_d^H & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}_{MP}^+ & (\mathbf{N}_d^H)_{MP}^+ \\ (\mathbf{N}_g)_{MP}^+ & 0 \end{pmatrix} \tag{4.22}$$

où l'on voit l'intérêt d'utiliser des bases orthonormales : dans ce cas $(\mathbf{N}_g)_{MP}^+ = \mathbf{N}_g^H$ *et* $(\mathbf{N}_d^H)_{MP}^+ = \mathbf{N}_d$

4.4.2 Calcul par blocs

Une autre technique repose sur une permutation préalable.

Proposition 4.4.5. *Si on découpe* $\mathbf{A}$ *par blocs*

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \qquad \textit{avec } \mathbf{A}_{11} \textit{ inversible et } \operatorname{rg}(\mathbf{A}) = \operatorname{rg}(\mathbf{A_{11}}) \tag{4.23}$$

On a alors

$$\begin{aligned} \mathbf{A}/\mathbf{A}_{11} &= \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} = \mathbf{0} \qquad \textit{(voir la notion de complément de Schur)} \\ \mathbf{N}_d &= \begin{pmatrix} -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{I}_{22} \end{pmatrix}, \quad \operatorname{Im}(\mathbf{A}) = \operatorname{Im}\begin{pmatrix} \mathbf{I}_{11} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} \end{pmatrix} \quad \textit{par exemple} \\ \mathbf{A}^+ &= \begin{pmatrix} \mathbf{A}_{11}^{-1} & 0 \\ 0 & 0 \end{pmatrix}, \quad \textit{par exemple} \end{aligned} \tag{4.24}$$

Dans le cas où $\mathbf{A}$ est une matrice carrée, alors la pseudo inverse précédente peut être approchée par pénalisation :

$$\mathbf{A}^+ = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} + \alpha\mathbf{I} \end{pmatrix}^{-1} \text{ avec } \alpha \text{ grand,} \tag{4.25}$$

Cette méthode a l'avantage d'être très peu invasive (il suffit d'accéder à quelques coefficients diagonaux d'une matrice creuse) et de rendre inversible la matrice à pseudo-inverser (qui peut donc utiliser une factorisation classique), le grand inconvénient de cette méthode est de détériorer fortement le conditionnement de l'opérateur.

D'après la proposition précédente, la connaissance d'une permutation (*ie* l'organisation en blocs avec le bloc $(1,1)$ inversible de rang égal à celui de la matrice) permet de connaître les noyaux (et donc potentiellement d'employer aussi les premières formules). Réciproquement, la connaissance des noyaux permet de trouver une permutation qui isole une matrice carrée de rang égal à celui de la matrice complète. En effet on voit sur (4.24) que le noyau est de rang plein sur les noeuds de type 2 ; cette propriété est résistante aux changements de base.

On raisonne dans le cas symétrique (sinon il faut distinguer permutation à gauche et à droite). Pour le noyau $\mathbf{N}$ donné, si la matrice carrée obtenue par extraction des lignes $(i_1, ...i_{\dim(\operatorname{Ker}_d(\mathbf{A}))})$ de $\mathbf{N}$ est inversible alors ces lignes peuvent être choisies pour définir le bloc 2 de la permutation. Mécaniquement cela correspond à bloquer des degrés de liberté de manière à supprimer les modes rigides.

On propose la technique suivante :

- on prend $p_1 = \operatorname{argmax}_i(|n_{i1}|)$ degré de liberté où le premier déplacement rigidifiant est le plus grand
- pour $j = 2 \cdots \dim(\operatorname{Ker}(\mathbf{A}))$ on met à jour $\mathbf{n}_j \leftarrow \mathbf{n}_j - \frac{n_{p_1 j}}{n_{p_1 1}}\mathbf{n}_1$ de manière à annuler la composante p_1 des autres déplacements rigides,
- on prend $p_2 = \operatorname{argmax}_i(|n_{i2}|)$ puisqu'on est sûr qu'il bloque le deuxième déplacement rigidifiant,
- on met à jour pour $j = 3 \cdots z$,$\mathbf{n}_j \leftarrow \mathbf{n}_j - \frac{n_{p_2 j}}{n_{p_2 2}}\mathbf{n}_2$ de manière à annuler la composante p_2 des autres déplacements rigides,
- on itère jusqu'à l'avant dernière colonne du noyau.

Cette méthode correspond à choisir des degrés de liberté « éloignés » pour bloquer la structure ce qui conduit à un meilleur conditionnement.

4.5 Obtention des informations relatives à la déficience de rang

On a vu qu'il était nécessaire de connaître soit une base du noyau soit une renumérotation permettant d'isoler un bloc de rang plein (et que ces deux connaissances étaient équivalentes). On donne ici des techniques qui permettent d'obtenir une de ces deux informations. On ne parle pas ici des factorisations QR, qui donnent toutes les informations souhaitables mais sont souvent trop lentes.

On a vu qu'il était possible de ramener un problème à un système carré et on s'intéresse donc à ce cas là (ceci dit ce n'est pas indispensable).

4.5.1 Méthode algébrique

Si on s'intéresse à la factorisation de Crout (ou *LDU*) d'une matrice, les singularités se manifestent par la présence de pivots nuls. Bien sûr la « nullité numérique » peut se révéler très difficile à définir ; on compare en général les pivots entre eux et si le conditionnement est mauvais l'écart entre les pivots peut être très faible. Il faut au moins utiliser un pivot partiel ; un pivot total est très sécurisant dans ce cas là.

Quand un pivot nul est détecté, une façon relativement peu invasive de travailler est d'annuler les coefficients de la ligne et la colonne du pivot et de mettre un 1 sur la diagonale du pivot. On peut alors finir la factorisation, il suffit ensuite de penser à annuler les composantes du second membre associées aux pivots nuls avant la descente/remontée. Une idée intéressante est de stocker $\mathbf{D}^{-1}$ (au lieu de $\mathbf{D}$) en mettant des 0 sur les pivots singuliers.

Une autre technique consiste à réaliser une pénalisation des pivots nuls : on remplace le pivot nul par un pivot énorme. Numériquement cette méthode est bien sûr moins propre).

Il faut remarquer qu'en pratique sauf numérotation et géométrie étranges, les pivots nuls se retrouvent à la fin de la matrice car les déplacements de solide rigide sont globaux sur toute la structure et c'est après avoir fait pivoter pratiquement tous les degrés de liberté qu'on arrive à annuler une colonne.

4.5.2 Méthode mécanique

On possède souvent une information sur les noyaux car ils sont issus de la physique, de la discrétisation, de la linéarisation. En mécanique on peut exploiter le fait que l'on sait *a priori* d'où provient le noyau. U

Une première technique consiste à utiliser une base candidate du noyau construite à partir des déplacements rigides de la structure supposée libre (trois translations, trois rotations infinitésimales). Cette base peut-être construite dès que l'on connaît la position des nœuds. On évalue alors cette base $\tilde{\mathbf{N}}$ ($n \times 6$ en 3D) en calculant $\tilde{\mathbf{M}} = \mathbf{A}\tilde{\mathbf{N}}$, il faut alors calculer le rang et une base de l'image de $\tilde{\mathbf{M}}$. Soit une matrice R (6×6), obtenue par orthogonalisation par exemple, telle que

$$\tilde{\mathbf{M}}\mathbf{R} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \end{pmatrix}, \quad \mathrm{rg}(\mathbf{M}) = \mathrm{rg}(\tilde{\mathbf{M}}) \tag{4.26}$$

alors les $(6 - \mathrm{rg}(\mathbf{M}))$ dernières colonnes de la matrice $\tilde{\mathbf{N}}\mathbf{R}$ forment une base du noyau de $\mathbf{A}$. Cette technique repose toujours sur l'évaluation d'un « zéro numérique » pour calculer le rang de $\tilde{\mathbf{M}}$.

Une technique plus sophistiquée consiste à analyser directement les conditions aux limites de Dirichlet. Pour cela on construit la matrice $\mathbf{C}$ ($n \times d$) dont chaque colonne de représente un degré de liberté bloqué. Alors $\mathbf{W} = \mathbf{C}^T\tilde{\mathbf{N}}$ représente le travail de les déplacements de solide rigide candidat dans les conditions aux limites imposées. Les déplacements réellement rigides sont les $\tilde{\mathbf{N}}\beta$ combinaisons linéaires de déplacements candidats où β est une base du noyau de $\mathbf{W}$. Pour trouver cette base, le plus élégant et le plus stable est d'utiliser la SVD. L'intérêt d'une telle méthode est de ne pas faire intervenir la rigidité de la structure et donc d'être beaucoup mieux conditionnée.

4.6 Complément de Schur généralisé

La notion de complément de Schur se généralise en considérant le cas où la matrice « dénominateur » n'est pas inversible (bien qu'il soit possible, moyennant des précautions, de travailler sur des matrices rectangulaires, on considère ici $\mathbf{A}$, $\mathbf{A}_{11}$ et $\mathbf{A}_{22}$ carrées).

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \tag{4.27}$$

on définit alors le complément de Schur généralisé :

$$\mathbf{A}/\mathbf{A}_{11} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{+}\mathbf{A}_{12} \tag{4.28}$$

La question fondamentale est de savoir à quelle condition le complément de Schur généralisé dépend du choix de l'inverse généralisée. On voit que la condition fondamentale est que $\mathrm{Im}(\mathbf{A_{12}}) \subset \mathrm{Im}(\mathbf{A_{11}})$ (comme ça le résultat de la multiplication $\mathbf{A}_{11}^{+}\mathbf{A}_{12}$ est déterminé de manière unique).

Proposition 4.6.1. *Si* $\mathbf{A}$ *hermitienne semi-définie positive.*

- *On a nécessairement* $\mathrm{Im}(\mathbf{A_{12}}) \subset \mathrm{Im}(\mathbf{A_{11}})$ *et le complément de Schur généralisé est bien défini. On a alors*

$$\mathrm{rg}(\mathbf{A}) = \mathrm{rg}(\mathbf{A_{11}}) + \mathrm{rg}(\mathbf{A}/\mathbf{A_{11}}) \tag{4.29}$$

- *On a une formule sur les pseudo inverses de Moore-Penrose dans le cas où* $\mathrm{rg}(\mathbf{A}) = \mathrm{rg}(\mathbf{A_{11}}) + \mathrm{rg}(\mathbf{A_{22}})$

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{+}_{MP} = \begin{pmatrix} \mathbf{A}_{11}{}^{+}_{MP} + (\mathbf{A}_{11})^{+}_{MP}\mathbf{A}_{12}(\mathbf{A}/\mathbf{A}_{11})^{+}_{MP}\mathbf{A}_{21}(\mathbf{A}_{11})^{+}_{MP} & -(\mathbf{A}_{11})^{+}_{MP}\mathbf{A}_{12}(\mathbf{A}/\mathbf{A}_{11})^{+}_{MP} \\ -(\mathbf{A}/\mathbf{A}_{11})^{+}_{MP}\mathbf{A}_{21}(\mathbf{A}_{11})^{+}_{MP} & (\mathbf{A}/\mathbf{A}_{11})^{+}_{MP} \end{pmatrix} \quad (4.30)$$

Chapitre 5

Solveurs directs

Les solveurs directs reposent sur l'observation que trois types de matrices sont simples à inverser : les matrices diagonales, les matrices triangulaires (par descente ou remontée), les matrices unitaires (il suffit de les transposer). Pour les matrices plus compliquées, on cherche à se ramener à une combinaison multiplicative des cas précédents. On parle de factorisation.

On considère le système $\mathbf{Ax} = \mathbf{b}$ dans $\mathbb{E}^n$.

5.1 Systèmes triangulaires

Les matrices triangulaires supérieures ou inférieures inversibles forment un groupe pour la multiplication. La résolution d'un système triangulaire est très simple (par descente ou remontée) et de complexité $O(n^2)$. Remarquons dès à présent qu'il est possible de travailler simultanément sur un ensemble de seconds membres (matrice rectangulaire avec relativement peu de colonnes, aussi appelée multivecteur). Le calcul du déterminant d'une matrice triangulaire est trivial (produit des coefficients diagonaux).

Concernant les problèmes de précision, on introduit le nombre $\gamma_n = nu/(1-nu)$ qui est très proche de nu sauf si la taille du système devient colossale. On note $|\mathbf{A}|$ la matrice obtenue en appliquant la valeur absolue (le module) à chacune des composante de $\mathbf{A}$.

On peut contrôler l'erreur rétrograde sous la forme $|\Delta\mathbf{A}| \leqslant \gamma_n|\mathbf{A}|$. Concernant l'erreur directe, on a :

$$||\Delta x||_\infty / \|x\|_\infty \leqslant \mathrm{cond}(T,x)\gamma_n/(1-\kappa_\infty(T)\gamma_n) \tag{5.1}$$

où $\mathrm{cond}(T,x) = \frac{\||T^{-1}|\,|T|\,|x|\|_\infty}{\|x\|_\infty} \leqslant \kappa_\infty(T)$.

5.2 Méthode de Gauss et méthodes dérivées

5.2.1 Gauss

Le principe de la méthode de Gauss est de trouver $\mathbf{M}$ triangulaire telle que $\mathbf{MAx} = \mathbf{Ux} = \mathbf{Mb}$ avec $\mathbf{U}$ triangulaire supérieure. C'est une méthode extrêmement générale, elle fait intervenir deux types matrices.

— matrices de permutation de ligne :

$$\mathbf{T}_{i,j} = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 0_{ii} & \cdots & \cdots & 1_{ij} & \\ & & \vdots & 1 & & \vdots & \\ & & \vdots & & 1 & \vdots & \\ & & 1_{ji} & \cdots & \cdots & 0_{jj} & \\ & & & & & & 1 \end{pmatrix} \tag{5.2}$$

Ces matrices ont un déterminant de (-1). Par convention on pose $\mathbf{T}_{i,i} = \mathbf{I}$ dont le déterminant est 1.

— matrices d'élimination de colonne :

$$\text{si } \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & & \cdots & & a_{n1} \\ 0 & \ddots & & & & \vdots \\ & \ddots & \ddots & & & \vdots \\ \vdots & & 0 & a_{jj} & \cdots & a_{jn} \\ & & \vdots & \vdots & & \vdots \\ 0 & & 0 & a_{nj} & \cdots & a_{nn} \end{pmatrix} \quad \text{avec } a_{jj} \neq 0 \tag{5.3}$$

$$\text{on pose } \gamma_{ij} = -a_{ij}/a_{jj} \qquad \text{soit } \mathbf{E}_j = \begin{pmatrix} 1 & 0 & \cdots & & & 0 \\ 0 & \ddots & \ddots & & & \\ \vdots & 0 & 1_{jj} & & & \\ & \vdots & \gamma_{(j+1)j} & \ddots & & \\ & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & 0 & \gamma_{nj} & 0 & 0 & 1 \end{pmatrix} \tag{5.4}$$

Alors $\mathbf{E}_j\mathbf{A}$ est une matrice dont on a recombiné les lignes $(j+1)$ à n pour faire disparaître les coefficients inférieurs de la colonne j. Les matrices $\mathbf{E}_j$ ont un déterminant de 1. En itérant on construit :

$$\mathbf{M} = \mathbf{E}_{n-1}\mathbf{T}_{n-1}\cdots\mathbf{E}_2\mathbf{T}_2\mathbf{E}_1\mathbf{T}_1, \text{ et alors } \mathbf{MA} = \mathbf{U} \tag{5.5}$$

Le rôle des matrices de permutation est de s'assurer que le pivot a_{ii} est non nul. Si tous ces coefficients sont « nuls » la matrice n'est pas inversible (on verra plus loin comment obtenir alors une base du noyau et une pseudo-inverse). Comme la matrice $\mathbf{M}$ a un déterminant de ± 1 (suivant la parité du nombre de pivotement), on a très simplement le déterminant de $\mathbf{A}$ égal au signe près à celui de $\mathbf{U}$.

Numériquement, il est conseillé de toujours prendre le plus grand coefficient disponible comme pivot. On distingue les stratégies de pivot partiel où l'on va chercher le coefficient sur la colonne j (ligne j à n) et les stratégies de pivot total où l'on va aussi chercher parmi les coefficients sur la ligne j (colonne j à n). Le pivot total revient à s'autoriser à multiplier la matrice à droite ce qui complexifie légèrement la méthode (surtout sa mise en œuvre).

Pour calculer complètement l'inverse, on peut utiliser la transformation de Gauss-Jordan :

$$\text{soit } \hat{\mathbf{E}}_j = \begin{pmatrix} 1 & & 0 & \gamma_{1j} & 0 & & 0 \\ 0 & \ddots & & \vdots & & & \\ & \ddots & \ddots & \gamma_{(j-1)j} & & & \\ \vdots & & 0 & 1_{jj} & 0 & & \\ & & \vdots & \gamma_{(j+1)j} & \ddots & & \\ & & \vdots & \vdots & 0 & \ddots & 0 \\ 0 & & 0 & \gamma_{nj} & 0 & 0 & 1 \end{pmatrix} \tag{5.6}$$

qui élimine entièrement la colonne j (sauf le terme diagonal).

La méthode du pivot de Gauss a une complexité en $O(n^3)$ sans compter la recherche des pivots qui n'est pas une opération de calcul mais une succession de conditions (qui ne correspondent pas au même type d'optimisations matérielles et logicielles).

5.2.2 Factorisation LU

Dans certains cas, il est possible de ne pas pivoter et d'obtenir directement $\mathbf{M}^{-1} = \mathbf{L}$, on a alors la factorisation $\mathbf{A} = \mathbf{LU}$ de la matrice (pour que la factorisation soit unique, on impose que la matrice $\mathbf{L}$ soit à diagonale unitaire). La condition de non-nécessité de pivotement est que tous les sous-déterminants soient non-nuls.

La condition ne garantit pas que les pivots « naturels » soient les meilleurs (les plus gros des coefficients disponibles), et il est plus prudent de pivoter malgré tout. Les matrices à diagonale dominante font exceptions puisque leurs pivots « naturels » sont automatiquement les plus gros coefficients disponibles.

Dans le cas tridiagonal, le calcul est particulièrement simple et la complexité devient de l'ordre de n. Les systèmes rectangulaires à sous-déterminants non-nuls peuvent aussi être factorisés sous forme LU (**L** ou **U** est de la même forme que **A** tandis que l'autre matrice est carrée).

Une variante de la factorisation LU est la factorisation LDU où **L** est triangulaire inférieur à diagonale unitaire, **D** est diagonale et **U** triangulaire supérieure à diagonale unitaire. Une telle variante permet de mettre en exergue les éléments diagonaux (qui doivent être inversés lors d'une résolution) pour traiter un éventuel problème mal posé.

Concernant la précision de la factorisation, on a le résultat suivant sur l'erreur rétrograde : $|\Delta\mathbf{A}| \leqslant \gamma_{3n}/(1-\gamma_n)|\mathbf{A}|$.

5.2.3 Factorisation de Crout

Si la matrice est LU-factorisable et qu'elle est symétrique, elle peut-être factorisée sous la forme de Crout : $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ avec **D** diagonale et **L** triangulaire inférieure à diagonale unitaire. Il est toujours recommandé de pivoter pour diminuer les erreurs d'arrondi, dans ce cas là il est nécessaire d'utiliser un pivot symétrique (multiplication à gauche par $\mathbf{T}_{ij}$ et à droite par $\mathbf{T}_{ij}^T = \mathbf{T}_{ij}$).

5.2.4 Factorisation de Cholesky

Si la matrice est symétrique définie positive alors elle peut être factorisée sous la forme $\mathbf{A} = \mathbf{L}\mathbf{L}^T$. On rappelle au passage qu'une matrice SPD a tous ses sous-déterminants strictement positifs.

Le nombre d'opérations est approximativement deux fois plus petit que pour une factorisation LU.

5.3 Orthogonalisation QR

Dans le cas d'une matrice carrée, la factorisation QR permet d'écrire une matrice **A** sous la forme du produit d'un matrice unitaire **Q** et d'une matrice triangulaire supérieure **R**.

Dans le cas d'une matrice rectangulaire on peut au choix obtenir **Q** unitaire $m \times m$ et **R** triangulaire supérieure $m \times n$ ou alors **Q** $m \times n$ avec des colonnes (si $m > n$) ou des lignes (si $m < n$) orthonormales entre elles et **R** triangulaire supérieure $n \times n$. Une application typique de cette factorisation est donc le calcul du rang d'une famille de vecteurs. On verra également qu'elle permet la résolution de problèmes au sens des moindres carrés. Pour les illustrations à venir on supposera la matrice carrée ou rectangulaire avec $n > m$ (cas assez récurent de la manipulation d'un petit paquet de très grands vecteurs).

On peut fournir plusieurs théorèmes d'orthogonalisation. On donne celui-ci : si **A** est une matrice carrée d'ordre n, il existe une matrice unitaire **Q** et une matrice triangulaire **R** telles que $\mathbf{A} = \mathbf{Q}\mathbf{R}$. On peut de plus s'arranger pour que tous les éléments diagonaux de **R** soient positifs (ou nuls). Si **A** est inversible, la factorisation est alors unique.

5.3.1 Version Gram-Schmidt

L'orthogonalisation la plus simple est celle de Gram-Schmidt, elle consiste à enchainer les manipulations de colonnes de la matrice. On pose et $r_{11} = \|\mathbf{a}_1\|_2$ et $\mathbf{q}_1 = \mathbf{a}_1/r_{11}$, si on suppose avoir construit les k premières colonnes de **Q** et le bloc $k \times k$ de **R** tels que

$$\begin{pmatrix} \vdots & \vdots & \vdots \\ \mathbf{a}_1, & \vdots & \mathbf{a}_k \\ \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \mathbf{q}_1, & \vdots & \mathbf{q}_k \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} r_{11} & \cdots & r_{1k} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{kk} \end{pmatrix} \qquad \text{avec } \mathbf{q}_i^H \mathbf{q}_j = \delta_{ij} \tag{5.7}$$

on calcule :

$$\begin{aligned} &\text{pour } i = 1 \cdots k \quad r_{i(k+1)} = \mathbf{q}_i^H \mathbf{a}_{k+1} \\ &\mathbf{q}_{k+1} = \mathbf{a}_{k+1} - \sum_i r_{i(k+1)} \mathbf{q}_i \\ &r_{(k+1)(k+1)} = \|\mathbf{q}_{k+1}\| \\ &\mathbf{q}_{k+1} \leftarrow \mathbf{q}_{k+1}/r_{(k+1)(k+1)} \end{aligned} \tag{5.8}$$

L'algorithme présenté ici est dit de Gram-Schmidt classique, il peut s'écrire avantageusement sous forme de produits de sous-blocs de matrice et est donc facilement optimisable (du point de vue exécution sur un ordinateur). Par contre l'algorithme n'est pas stable. Si $\mathbf{Q}$ et $\mathbf{R}$ sont les matrices obtenues, on a $\|\mathbf{A}-\mathbf{QR}\|$ petit, mais $\|\mathbf{Q}^T\mathbf{Q}-\mathbf{I}\|$ devient rapidement grand, autrement dit on perd l'orthogonalité. Cela pose problème si l'on veut se servir de la factorisation pour résoudre un système

Pour résoudre ce problème, on peut utilisée l'algorithme de Gram-Schmidt modifié, beaucoup plus stable (en fait très proche de Householder présenté plus bas) mais beaucoup plus coûteux :

$$\begin{array}{l}\mathbf{q}_{k+1}=\mathbf{a}_{k+1}\\ \text{pour } i=1..(k+1)\quad r_{i(k+1)}=\mathbf{q}_i^H\mathbf{q}_{k+1},\quad \mathbf{q}_{k+1}\leftarrow\mathbf{q}_{k+1}-r_{i(k+1)}\mathbf{q}_i\\ r_{(k+1)(k+1)}=\|\mathbf{q}_{k+1}\|\\ \mathbf{q}_{k+1}\leftarrow\mathbf{q}_{k+1}/r_{(k+1)(k+1)}\end{array} \tag{5.9}$$

Même améliorée, la méthode de Gram-Schmidt est simple à coder, à optimiser et surtout permet d'accéder à $\mathbf{Q}$ (si seul $\mathbf{R}$ est recherché Householder est plus rapide).

5.3.2 Version Householder

L'intérêt de la méthode de Householder est de faire intervenir des transformations unitaires dont on a vu qu'elles ne dégradaient pas le conditionnement κ_2. Elle est donc réputée beaucoup plus stable qu'une technique de Gauss pour résoudre un système mais elle fait intervenir environ deux fois plus d'opérations. La méthode repose sur l'emploi des matrices de réflexion dites de Householder.

On appelle matrice de réflexion de Householder la matrice $n\times n$ $\mathbf{H}$:

$$\mathbf{H}_\mathbf{v}=\mathbf{I}-2\frac{\mathbf{v}\mathbf{v}^H}{\mathbf{v}^H\mathbf{v}} \tag{5.10}$$

$\mathbf{v}$ est le vecteur de Householder. $\mathbf{P}$ est unitaire et hermitienne, elle réalise une réflexion par rapport à l'hyperplan $\text{vect}(\mathbf{v})^\perp$. Cette matrice est obtenue à partir de l'identité à l'aide d'une perturbation de rang 1.

Le théorème utile est le suivant : Soit $\mathbf{a}$ un vecteur de $\mathbb{C}^n$ tel que $a_1\neq 0$ et $\sum_{i=2}^n|a_i|>0$. Il existe deux matrices de Householder $\mathbf{H}$ telles que les $(n-1)$ dernières composantes du vecteur $\mathbf{Ha}$ soit nulles. De façon plus précise, soit $\alpha\in\mathbb{R}$ tel que $a_1=e^{i\alpha}|a_1|$, alors si $\mathbf{e}_1$ est le premier vecteur de base de $\mathbb{C}^n$:

$$\mathbf{H}_{(\mathbf{a}+\|\mathbf{a}\|_2e^{i\alpha}\mathbf{e}_1)}\mathbf{a}=-\|\mathbf{a}\|_2\mathbf{e}_1,\qquad \mathbf{H}_{(\mathbf{a}-\|\mathbf{a}\|_2e^{i\alpha}\mathbf{e}_1)}\mathbf{a}=\|\mathbf{a}\|_2\mathbf{e}_1 \tag{5.11}$$

En pratique, on calcule $\|\mathbf{a}\|_2$ puis $\mathbf{v}=(\mathbf{a}\pm\|\mathbf{a}\|_2e^{i\alpha}\mathbf{e}_1)$ et le nombre $\frac{\mathbf{v}^H\mathbf{v}}{2}=\|\mathbf{a}\|_2(\|\mathbf{a}\|_2\pm|a_1|)$. Le calcul de l'image d'un vecteur $\mathbf{x}$ se fait de la façon suivante :

$$\mathbf{H}_v\mathbf{x}=\mathbf{x}-\frac{\mathbf{v}^H\mathbf{x}}{\mathbf{v}^H\mathbf{v}/2}\mathbf{v} \tag{5.12}$$

Dans le cas réel le *signe est choisi* de manière à éviter un dénominateur « trop petit » donc on prend $\mathbf{v}=(\mathbf{a}+\text{signe}(a_1)\|\mathbf{a}\|_2\mathbf{e}_1)$.

La méthode de Householder consiste donc à calculer $\mathbf{A}_k=\mathbf{H}_{k-1}\cdots\mathbf{H}_1\mathbf{A}$ avec :

$$\mathbf{A}_k=\begin{pmatrix}\times&\times&\times&\times&\times&\times\\ &\times&\times&\times&\times&\times\\ &&\times_{kk}&\times&\times&\times\\ &&\times&\times&\times&\times\\ &&\times&\times&\times&\times\\ &&\times_{nk}&\times&\times&\times\end{pmatrix} \tag{5.13}$$

Si besoin on pratique une permutation de lignes (permutation à droite) de manière à rendre le coefficient kk non-nul. On extrait de $\mathbf{A}_k$ le vecteur $\tilde{\mathbf{a}}_k$ de dimension $(n-k+1)$ de coefficients a_{ik} $(k\leqslant i\leqslant n)$. On choisit le vecteur $\tilde{\mathbf{v}}_k$ tel que $\mathbf{H}_{\tilde{\mathbf{v}}_k}\mathbf{v}_k$ ait toutes ses composantes nulles sauf la première et on pose :

$$\mathbf{H}_k=\begin{pmatrix}\mathbf{I}_{k-1}&\mathbf{0}\\ \mathbf{0}&\mathbf{H}_{\tilde{\mathbf{v}}_k}\end{pmatrix} \tag{5.14}$$

Il est possible de représenter le produit de r matrices de Householder sous forme bloc :

$$\mathbf{Q} = \mathbf{I} + \mathbf{W}\mathbf{Y}^H \tag{5.15}$$

où $\mathbf{W}$ et $\mathbf{Y}$ sont des matrices $n \times r$. Le mécanisme est le suivant :

$$\mathbf{Q}\mathbf{H}_\mathbf{v} = \mathbf{I} + \left(\mathbf{W}, \quad -\tfrac{2}{\mathbf{v}^H\mathbf{v}}\mathbf{Q}\mathbf{v}\right)\left(\mathbf{Y}, \quad \mathbf{v}\right)^H \tag{5.16}$$

5.3.3 Version Givens

A l'instar de Householder, l'orthogonalisation de Givens fait intervenir des transformations unitaires et donc ne détériore pas le conditionnement κ_2. La méthode repose sur l'emploi des matrices de rotations dites de Givens qui sont des perturbations de rang 2 de l'identité. Ces matrices permettent d'intervenir plus spécifiquement sur certains coefficients d'une matrice. Par simplicité, on se place dans $\mathbb{R}$. On pose :

$$\mathbf{G}_{ij}(\theta) = \begin{pmatrix} \mathbf{I}_{i-1} & & & & \mathbf{0} \\ & \cos(\theta)_{ii} & \mathbf{0} & \sin(\theta)_{ij} & \\ & \mathbf{0} & \mathbf{I}_{n-2} & \mathbf{0} & \\ & -\sin(\theta)_{ji} & \mathbf{0} & \cos(\theta)_{jj} & \\ \mathbf{0} & & & & \mathbf{I}_{n-j} \end{pmatrix} \tag{5.17}$$

Si on calcule le produit suivant :

$$\mathbf{Y} = \mathbf{G}_{ik}(\theta)^T\mathbf{X}; \qquad y_{qr} = \begin{cases} \cos(\theta)x_{ir} - \sin(\theta)x_{jr} & q = i, \ \forall r \\ \sin(\theta)x_{ir} + \cos(\theta)x_{jr} & q = j, \ \forall r \\ x_{qr} & q \neq i, j, \ \forall r \end{cases} \tag{5.18}$$

L'idée derrière cela est de calculer θ pour annuler des coefficients. Par exemple si on veut annuler y_{qr} (ici les deux indices sont fixés), on prend i tel que $x_{ir} \neq 0$ et on pose

$$\cos(\theta) = \frac{x_{ir}}{\sqrt{x_{ir}^2 + x_{qr}^2}}, \qquad \sin(\theta) = \frac{x_{qr}}{\sqrt{x_{ir}^2 + x_{qr}^2}} \tag{5.19}$$

Pour minimiser les erreurs d'arrondi, si a et b sont donnés et que l'on cherche $c = \cos(\theta)$ et $s = \sin(\theta)$ tels que

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix} \tag{5.20}$$

alors

$$\begin{array}{ll} \text{si } b = 0, & c = 1, \ s = 0 \\ \text{si } |b| > |a|, & \tau = -a/b, \ s = 1/\sqrt{1+\tau^2}, \ c = s\tau \\ \text{si } |b| \leqslant |a|, & \tau = -b/a, \ c = 1/\sqrt{1+\tau^2}, \ c = c\tau \end{array} \tag{5.21}$$

Une utilisation typique des rotations de Givens (le tilde indique une modification du coefficient d'une étape à l'autre) :

$$\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \xrightarrow{(3,1)} \begin{pmatrix} \times & \times & \times \\ \tilde{\times} & \tilde{\times} & \tilde{\times} \\ 0 & \tilde{\times} & \tilde{\times} \end{pmatrix} \xrightarrow{(2,1)} \begin{pmatrix} \tilde{\times} & \tilde{\times} & \tilde{\times} \\ 0 & \tilde{\times} & \tilde{\times} \\ 0 & \times & \times \end{pmatrix} \xrightarrow{(3,2)} \begin{pmatrix} \times & \times & \times \\ 0 & \tilde{\times} & \tilde{\times} \\ 0 & 0 & \tilde{\times} \end{pmatrix} \tag{5.22}$$

On verra une application typique dans le solveur GMRes pour résoudre un système de Hessemberg.

5.3.4 Factorisation QR et systèmes rectangulaires

Les factorisation QR se prêtent bien au calcul de systèmes rectangulaires car la préservation de la norme permet de facilement intégrer les régularisations présentées précédemment.

Si on considère un système sur déterminé $\mathbf{A}\mathbf{x} = \mathbf{b}$ avec $\mathbf{A} \in \mathbb{E}^{m\times n}$ et $m > n$. On a :

$$\begin{aligned} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 &= \left\| \mathbf{b} - \begin{pmatrix} \mathbf{Q}_1 & \mathbf{Q_2} \end{pmatrix} \begin{pmatrix} \mathbf{R}_1 \\ 0 \end{pmatrix} \right\|_2^2 \\ &= \left\| \begin{pmatrix} \mathbf{Q}_1^H\mathbf{b} - \mathbf{R}_1\mathbf{x} \\ \mathbf{Q}_2^H\mathbf{b} \end{pmatrix} \right\|_2^2 = \left\|\mathbf{Q}_1^H\mathbf{b} - \mathbf{R}_1\mathbf{x}\right\|_2^2 + \left\|\mathbf{Q}_2^H\mathbf{b}\right\|_2^2 \end{aligned} \tag{5.23}$$

où l'on voit que le minimum est égal à $\left\|\mathbf{Q}_2^H\mathbf{b}\right\|_2^2$ (qui mesure l'écart du second membre à l'image de $\mathbf{A}$) et est atteint pour $\mathbf{R}_1\mathbf{x} = \mathbf{Q}_1^H\mathbf{b}$ (petit système triangulaire).

Concernant les systèmes sous-déterminés $m < n$, on peut utiliser la factortisation QR de $\mathbf{A}^T$. On a :

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{R}^\mathbf{T}\mathbf{Q}^T\mathbf{x} &= \mathbf{b} \end{aligned} \tag{5.24}$$

Si on cherche $\mathbf{x}$ sous la forme $\mathbf{Q}\mathbf{y}+\mathbf{z}$ avec $\mathbf{Q}^T\mathbf{z} = 0$, on a $\|\mathbf{x}\|_2^2 = \|\mathbf{y}\|_2^2 + \|\mathbf{z}\|_2^2$. Donc $\mathbf{x} = \mathbf{Q}\mathbf{R}^{-T}\mathbf{b}$ est la solution de norme minimale.

Chapitre 6

Solveurs itératifs

Les méthodes directes atteignent leurs limites dans le cas de problèmes à grande largeur de bande et grand nombre de degrés de liberté. Néanmoins les architectures de calcul modernes multicœurs permettent d'envisager des performances très intéressantes pour ces méthodes pour des problèmes à plusieurs millions d'inconnues. Les solveurs itératifs reposent sur l'introduction d'une tolérance sur le résultat espéré, il doit s'approcher suffisamment près (dans un sens contrôlé) de la solution, ils sont potentiellement beaucoup plus rapide que les solveurs directs (qui obtiennent la solution en temps fixé par le profil de la matrice) mais beaucoup plus sensible au conditionnement des opérateurs puisque aux erreurs inévitables d'arrondi sont ajoutés des erreurs d'approximation dues aux méthodes même.

Les solveurs itératifs sont toujours abordés dans les bouquins de méthodes numériques (cf références de la première partie). La littérature relative aux solveurs de Krylov ne cesse de s'accroître, pour l'instant un point d'entrée incontournable est le *iterative method for sparse linear systems* de Y. Saad (SIAM). Pour une mise en oeuvre rapide, le *Template for ...* est une référence intéressante.

6.1 Notations et contrôle de l'erreur

On résout toujours $\mathbf{A}\mathbf{x} = \mathbf{b}$. On note $\mathbf{x}_i$ la i-ème approximation, $\delta_i = \mathbf{x} - \mathbf{x}_i$ l'erreur et $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\mathbf{x}_i = \mathbf{A}(\mathbf{x} - \mathbf{x}_i) = \mathbf{A}\delta_i$ le résidu associé. L'initialisation joue un rôle un peu particulier, et on décomposera souvent $\mathbf{x}_m = \mathbf{x}_0 + \tilde{\mathbf{x}}_m$, $\mathbf{r}_m = \mathbf{r}_0 - \mathbf{A}\tilde{\mathbf{x}}_m$. On fait souvent appel à des groupes de vecteurs, on les notera comme des matrices $\mathbf{R}_m = \left(\mathbf{r}_0 \cdots \mathbf{r}_{m-1}\right)$ (l'indice fait donc référence au nombre de colonnes).

Le principe même d'un solveur itératif est d'être arrêté dès qu'un critère mesurable est satisfait. Le principal critère est lié au résidu, on itère tant que $\|\mathbf{r}_i\| > \varepsilon\|\mathbf{b}\|$ avec ε choisi à partir de (i) expérience utilisateur, (ii) si possible lien avec une erreur issue d'une autre approximation (par exemple erreur de discrétisation d'une méthode élément fini – rien ne sert d'un point de vue mécanique de faire une résolution précise sur un maillage ultra grossier), (iii) l'erreur d'un solveur de niveau supérieur (cas des méthodes Newton inexact où on se permet de faire moins bien converger le solveur tangent quand le résidu non-linéaire est grand). Bien sûr, on a $\|delta_i\| \leqslant \|\mathbf{A}^{-1}\|\|\mathbf{r}_i\|$ et le résidu est une bonne information que si l'inverse du problème est de norme raisonnable.

Un autre critère possible est la "stagnation" du solveur $\|\mathbf{x}_{i+1} - \mathbf{x}_i\|$. Dans certains cas on peut lier cette quantité à l'erreur.

6.2 Méthodes de stationnarité

Ces méthodes reposent sur le partitionnement de la matrice $\mathbf{A} = \mathbf{M} - \mathbf{N}$ avec $\mathbf{M}$ facile à « inverser ». On réécrit le système :

$$\mathbf{x} = \mathbf{M}^{-1}\mathbf{N}\mathbf{x} + \mathbf{M}^{-1}\mathbf{b} \tag{6.1}$$

On utilise un schéma de type point fixe :

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{M}^{-1}\mathbf{N}\mathbf{x}_i + \mathbf{M}^{-1}\mathbf{b} \\ \delta_{i+1} &= \mathbf{M}^{-1}\mathbf{N}\delta_i \end{aligned} \tag{6.2}$$

qui converge dès que $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$.

Un premier partitionnement consiste à choisir $\mathbf{M} = \mathbf{I}/\omega$ et $\mathbf{N} = \mathbf{I}/\omega - \mathbf{A}$, ce qui conduit aux itérations de Richardson :

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \omega\mathbf{r}_i = (\mathbf{I} - \omega\mathbf{A})\mathbf{x}_i + \omega\mathbf{b} \tag{6.3}$$

Si on écrit $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$ avec $\mathbf{D}$ diagonale $\mathbf{E}$ strictement triangulaire inférieure et $\mathbf{F}$ strictement triangulaire supérieure. On distingue les méthodes suivantes :

— Jacobi : $\mathbf{M} = \mathbf{D}$ et $\mathbf{N} = \mathbf{E} + \mathbf{F}$,
— Gauss-Seidel : $\mathbf{M} = \mathbf{D} - \mathbf{E}$ et $\mathbf{N} = \mathbf{F}$,
— Gauss-Seidel rétrograde (*backward*) : $\mathbf{M} = \mathbf{D} - \mathbf{F}$ et $\mathbf{N} = \mathbf{E}$,
— Gauss-Seidel symétrique : enchainement d'un Gauss-Seidel classique et d'un Gauss-Seidel rétrograde,
— Sur-relaxation, ω paramètre réel : $\mathbf{M} = \mathbf{D} - \omega\mathbf{F}$ et $\mathbf{N} = (\frac{1}{\omega} - 1)\mathbf{D} + \mathbf{E}$ (en Anglais on parle de *successive overrelaxation* (SOR)),
— Sur-relaxation rétrograde et symétrique (SSOR).

Dans les méthodes de Richardson et de sur-relaxation on distingue la zone où ω rend la méthode convergente, et la valeur qui rend la convergence optimale (plus petit rayon spectral).

On voit que ces systèmes sont équivalents à résoudre :

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{6.4}$$

on interprète alors $\mathbf{M}^{-1}$ comme un préconditionneur (à gauche) : un opérateur qui ne modifie pas la solution du problème mais donne un système équivalent mieux conditionné. Ici vue la condition de convergence, il s'agit de rendre le rayon spectral de la matrice d'itération $(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A})$ le plus petit possible (et impérativement inférieur à 1).

Dans les méthodes de stationnarité, on a les propriétés suivantes :

$$\begin{aligned} \mathbf{x}_{k+1} - \mathbf{x}_k &= \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_k) = \mathbf{M}^{-1}\mathbf{r}_k \\ &= (\mathbf{I} - \mathbf{M}^{-1}\mathbf{N})\delta_k \end{aligned} \tag{6.5}$$

Autrement dit le résidu préconditionné correspond à la stagnation du solveur. Si $\mathbf{M}^{-1}\mathbf{N}$ est trop proche de l'identité (mauvais rayon spectral), la stagnation donne une mauvaise information sur la convergence de la méthode.

Il existe énormément de résultats de convergence de ces méthodes, selon le signe des coefficients des matrices $\mathbf{M}$, $\mathbf{N}$ et $\mathbf{A}^{-1}$, le caractère diagonal-dominant, le caractère symétrique défini positif, le profil de la matrice, etc.

6.3 Méthodes de projection

6.3.1 Principes

Soit $\mathbb{V}$ et $\mathbb{W}$ deux sous-espaces de dimension m. Une technique de projection sur $\mathbb{V}$ (espace d'approximation) orthogonalement à $\mathbb{W}$ (espace de contrainte), en connaissance d'une initialisation $\mathbf{x}_0$, consiste à chercher l'approximation $\mathbf{x}_m$ telle que :

$$\begin{aligned} \mathbf{x}_m &\in \mathbf{x}_0 + \mathbb{V} \\ \mathbf{r}_m &\perp \mathbb{W} \end{aligned} \tag{6.6}$$

Ce cadre de recherche (espace d'approximation/espace de contrainte) très classique forme les conditions de Petrov-Galerkin ; dans le cas d'une projection orthogonale ($\mathbb{W} = \mathbb{V}$) on parle simplement de conditions de Galerkin.

Matriciellement la méthode s'écrit $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}\mathbf{y}$, et $0 = \mathbf{W}^T\mathbf{r}_m = \mathbf{W}^T(\mathbf{r}_0 - \mathbf{A}\tilde{\mathbf{x}}_m)$ soit

$$\begin{aligned} \mathbf{x}_m &= \mathbf{x}_0 + \mathbf{V}(\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0 \\ \mathbf{r}_m &= (\mathbf{I} - \mathbf{A}\mathbf{V}(\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T)\mathbf{r}_0 \end{aligned} \tag{6.7}$$

Sous certaines conditions la matrice $(\mathbf{W}^T\mathbf{A}\mathbf{V})$ est inversible ce qui rend l'approximation bien définie. Les algorithmes basés sur un principe de projection évitent en général de devoir inverser réellement cette matrice.

Proposition 6.3.1. *On a deux résultats importants :*
— *Si* $\mathbf{A}$ *est symétrique définie positive et* $\mathbb{V} = \mathbb{W}$ *alors* $\mathbf{x}_m$ *minimise* $\|\delta_m\|_A$ *sur* $\mathbb{K}$.

— *Si* $\mathbb{W} = \mathbf{A}\mathbb{V}$ *alors* $\mathbf{x}_m$ *minimise* $\|\mathbf{r}_m\|_2$ *sur* $\mathbb{K}$.

Démonstration. Avec les notations précédentes,
— On a $\|\mathbf{x} - \mathbf{x}_0 - \mathbf{V}\mathbf{y}\|_A^2 = \|\mathbf{x} - \mathbf{x}_0\|_A^2 + \|\mathbf{V}\mathbf{y}\|_A^2 - 2(\mathbf{V}\mathbf{y})^T\mathbf{A}(\mathbf{x} - \mathbf{x}_0)$, en utilisant $\mathbf{A}(\mathbf{x} - \mathbf{x}_0) = \mathbf{r}_0$ puis en minimisant par rapport à $\mathbf{y}$ on obtient la formule (6.7) avec $\mathbf{W} = \mathbf{V}$.
— On a $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0 - \mathbf{A}\mathbf{V}\mathbf{y}\|_2^2 = \|\mathbf{r}_0\|_2^2 + \|\mathbf{A}\mathbf{V}\mathbf{y}\|_2^2 - 2(\mathbf{A}\mathbf{V}\mathbf{y})^T\mathbf{r}_0$ en minimisant par rapport à à $\mathbf{y}$ on obtient la formule (6.7) avec $\mathbf{W} = \mathbf{A}\mathbf{V}$.

□

6.3.2 Algorithmes 1D

On choisit ici de ne retenir que des espaces de dimension 1 réinitialisés à chaque itération. Autrement dit ces algorithmes ont une mémoire minimale : d'une itération à l'autre on se souvient uniquement de l'approximation trouvée (et du résidu associé).

Descente optimale – *Steepest descent*

$\mathbf{A}$ est SPD. On choisit ici de faire variation $\mathbf{x}$ dans la direction qui maximise (localement) la variation de la $\mathbf{A}$-norme de l'erreur. On se place donc dans le cadre $\mathbb{V} = \text{vect}(\mathbf{r}) = \mathbb{W}$. La méthode converge pour toute matrice

Algorithm 1: Steepest descent

Calculer $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ et $\mathbf{p} = \mathbf{A}\mathbf{r}$;
for *convergence* **do**
  $\alpha = \mathbf{r}^T\mathbf{r}/\mathbf{p}^T\mathbf{r}$;
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{r}$;
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{p}$;
end

SPD. Le taux de convergence est donné par :

$$\|\delta_{k+1}\|_A \leqslant \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\|\delta_k\|_A \tag{6.8}$$

Résidu minimal – MR

On se place ici dans le cas d'une matrice définie positive (*ie* la partie symétrique $\mathbf{A}^s$ de la matrice est SPD). On se place ici dans le cadre $\mathbb{V} = \text{vect}(\mathbf{r})$ et $\mathbb{W} = \text{vect}(\mathbf{A}\mathbf{r})$ et on minimise donc la 2-norme du résidu. La

Algorithm 2: Résidu minimal

Calculer $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ et $\mathbf{p} = \mathbf{A}\mathbf{r}$;
for *convergence* **do**
  $\alpha = \mathbf{p}^T\mathbf{r}/\mathbf{p}^T\mathbf{p}$;
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{r}$;
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{p}$;
end

méthode converge, le taux est donné par :

$$\|\mathbf{r}_{k+1}\|_2 \leqslant \left(1 - \frac{\lambda_{\min}(\mathbf{A}^s)^2}{\sigma(\mathbf{A})^2}\right)^{1/2}\|\mathbf{r}_k\|_2 \tag{6.9}$$

Descente optimale pour le problème normal

On se place ici dans le cas d'une matrice inversible. On applique une *steepest descent* sur l'équation normale, $\mathbb{V} = \text{vect}(\mathbf{A}^T\mathbf{r})$ et $\mathbb{W} = \text{vect}(\mathbf{A}\mathbf{r})$ et on minimise donc la 2-norme du résidu dans la direction optimale. La matrice du problème normal est SPD donc la convergence est guidée par les résultats de l'algorithme de *steepest descent*.

Algorithm 3: Descente optimale sur le problème normal

Calculer $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$;
for *convergence* **do**
  $\mathbf{v} = \mathbf{A}^T\mathbf{r}$;
  $\mathbf{w} = \mathbf{A}\mathbf{v}$;
  $\alpha = \mathbf{v}^T\mathbf{v}/\mathbf{w}^T\mathbf{w}$;
  $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{v}$;
  $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{w}$;
end

6.4 Solveurs de Krylov

Les algorithmes de Krylov sont des méthodes de projection sur des espaces de dimension croissante. On définit le sous-espace de Krylov $\mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) = \text{vect}\left(\mathbf{r}_0, \cdots, \mathbf{A}^{m-1}\mathbf{r}_0\right)$ dans lequel la solution est recherchée (modulo une initialisation). L'espace de contrainte $\mathbb{L}_m$ permettra de définir les différentes méthodes. Les implémentations permettent de rendre la recherche de la m-ème approximation peu coûteuse malgré l'optimisation vis-à-vis d'un espace de taille croissante.

Du fait de la forme de l'espace de Krylov, on voit que l'on cherche la solution sous la forme $\mathbf{x} = \mathbf{x}_0 + q(\mathbf{A})\mathbf{r}_0$ avec q un polynôme de degré $m-1$. On appelle le grade du vecteur $\mathbf{r}_0$ le rang m à partir duquel l'espace de Krylov devient invariant : $\mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) = \mathbb{K}_\mu(\mathbf{A}, \mathbf{r}_0),\ \mu \geqslant m$. C'est aussi le degré du polynôme minimal de $\mathbf{A}$ associé à $\mathbf{r}_0$ (le plus « petit » polynôme tel que $q(\mathbf{A})\mathbf{r}_0 = 0$).

6.4.1 Algorithmes basés sur la méthode d'Arnoldi

La méthode d'Arnoldi permet de réduire une matrice sous forme Hessemberg supérieure à l'aide de transformation unitaire. La base des $\mathbf{V}_m$ est 2-orthonormale et vérifie les propriétés suivantes :

Algorithm 4: Méthode d'Arnoldi

Choisir $\mathbf{v}_1$ tel que $\|\mathbf{v}_1\|_2 = 1$;
for *j=1,...,m* **do**
  $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$;
  $h_{ij} = \mathbf{v}_i^T\mathbf{w}_j$ pour $i = 1, ..., j$;
  $\mathbf{w}_j \leftarrow \mathbf{w}_j - \sum_i h_{ij}\mathbf{v}_i$;
  $h_{(j+1)j} = \|\mathbf{w}_j\|_2$ Si $h_{(j+1)j} = 0$ stop;
  $\mathbf{v}_{j+1} = \mathbf{w}_j/h_{(j+1)j}$;
end

$$\begin{aligned} \mathbf{A}\mathbf{V}_m &= \mathbf{V}_{m+1}\bar{\mathbf{H}}_m \\ \mathbf{A}\mathbf{V}_m &= \mathbf{V}_m\mathbf{H}_m + h_{(m+1)m}\mathbf{v}_{m+1}\mathbf{e}_m^T \\ \mathbf{V}_m^T\mathbf{A}\mathbf{V}_m &= \mathbf{H}_m \end{aligned} \tag{6.10}$$

La matrice $\mathbf{H}_m$ est carrée $m \times m$ Hessemberg supérieure (non-nulle dans le triangle supérieur et la première sous-diagonale). La matrice $\bar{\mathbf{H}}_m$ est rectangulaire ($m+1$ lignes, m colonnes), sur la dernière ligne, seul le dernier coefficient est non-nul.

Les méthodes de résolution vont consister à chercher $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$ en initialisant la construction de la base par le vecteur $\mathbf{v}_1 = \frac{\mathbf{r}_0}{\|\mathbf{r}_0\|}$.

Méthode FOM

La méthode FOM (*full orthogonalisation method*) consiste à choisir pour espace de projection $\mathbb{L}_m = \mathbb{K}_m$.

$$\begin{aligned} \mathbf{V}_m^T\mathbf{A}\mathbf{x}_m = \mathbf{V}_m^T\mathbf{r}_0 &= \|\mathbf{r}_0\|\mathbf{e}_1 \\ \mathbf{x}_m &= \mathbf{x}_0 + \|\mathbf{r}_0\|\mathbf{V}_m\mathbf{H}_m^{-1}\mathbf{e}_1 \end{aligned} \tag{6.11}$$

Cette méthode est très conceptuellement très simple, elle est à la base de beaucoup d'autres et beaucoup d'algorithmes peuvent être décrits comme une FOM appliquée à un problème modifié. Néanmoins cette méthode est très peu employée car elle présente d'importantes lacunes pratiques, notamment il est impossible de connaitre la qualité de l'approximation sans faire explicitement le calcul de $\mathbf{x}_m$ (pour en déduire la norme du résidu $\mathbf{r}_m$). Or l'inversion de $\mathbf{H}_m$ reste une opération couteuse. Pour les systèmes généraux, on préfère donc en général l'utilisation de GMRes ; pour les systèmes symétriques, FOM est équivalente à un gradient conjugué dont la mise en œuvre est très efficace.

On remarque au passage que comme $\mathbf{r}_{m+1} \perp \mathbb{K}_m$, l'espace de Krylov est également engendré par la suite des résidus : $\mathbb{K}_{m+1} = \text{vect}(\mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_m)$. Plus précisément, on a

$$\begin{aligned} \mathbf{r}_m &= \mathbf{r}_0 - \|\mathbf{r}_0\|\mathbf{A}\mathbf{V}_m\mathbf{H}_m^{-1}\mathbf{e}_1 \\ &= \|\mathbf{r}_0\|(\mathbf{v}_1 - \mathbf{V}_m\mathbf{H}_m + h_{(m+1)m}\mathbf{v}_{m+1}\mathbf{e}_m^T)\mathbf{H}_m^{-1}\mathbf{e}_1 \\ &= \|\mathbf{r}_0\|h_{(m+1)m}(\mathbf{e}_m^T\mathbf{H}_m^{-1}\mathbf{e}_1)\mathbf{v}_{m+1} \end{aligned} \tag{6.12}$$

Le résidu à l'itération m est colinéaire au $m+1$ vecteur de la base d'Arnoldi.

Méthode GMRes

GMRes est une méthode dont il existe des mises en oeuvre très astucieuses. La méthode repose sur le choix $\mathbb{L}_m = \mathbf{A}\mathbb{K}_m$ qui est équivalent à choisir de minimiser la norme du résidu à chaque itération. On a, en utilisant l'orthogonalité des colonnes de $\mathbf{V}_{m+1}$:

$$\begin{aligned} \mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m &= \mathbf{r}_0 - \mathbf{A}\mathbf{V}_m\mathbf{y}_m \\ &= \|\mathbf{r}_0\|\mathbf{v}_1 - \mathbf{V}_{m+1}\bar{\mathbf{H}}_m\mathbf{y}_m \\ &= \mathbf{V}_{m+1}(\|\mathbf{r}_0\|\mathbf{e}_1 - \bar{\mathbf{H}}_m\mathbf{y}_m) \\ \|\mathbf{r}_m\|_2 &= \|\|\mathbf{r}_0\|\mathbf{e}_1 - \bar{\mathbf{H}}_m\mathbf{y}_m\|_2 \end{aligned} \tag{6.13}$$

L'intérêt de GMRes est qu'il n'est pas nécessaire de minimiser (6.13) pour obtenir un indicateur de convergence. Il est en effet possible de connaître la norme du résidu sans calculer explicitement la solution à chaque itération. Le principe repose sur l'utilisation de rotations de Givens pour trianguläriser la matrice $\bar{\mathbf{H}}_m$:

- à chaque itération on rajoute une colonne à $\hat{\mathbf{H}}_m$
- on applique à cette colonne les rotations précédemment déterminée (la matrice obtenue n'a qu'un coefficient non triangulaire),
- on détermine l'angle qui permet de trianguläriser la matrice (supprimer le seul coefficient hors triangle)
- on met à jour le second membre en appliquant la nouvelle rotation (le second membre est un vecteur de taille $m+1$)

A la fin du processus on a face à face une matrice rectangulaire (triangulaire supérieure avec la dernière ligne de zéros) et un vecteur $m+1$. Puisqu'on appliqué des transformations unitaires, la norme du second membre est inchangée et le $(m+1)^{eme}$ coefficient est égal à la norme du résidu après inversion (virtuelle) de la matrice triangulaire.

Ainsi à chaque itération la matrice est agrandie, triangularisée, la norme du résidu est évaluée, quand elle est suffisamment petite on stoppe les itérations et on inverse le système triangulaire pour avoir la solution.

6.4.2 Algorithmes basés sur la méthode de Lanczos symétrique

La méthode de Lanczos peut être vue comme une simplification de la méthode d'Arnoldi dans le cas symétrique puisqu'alors la matrice $\mathbf{H}_m$ est tridiagonale symétrique. L'orthogonalisation peut alors se faire selon une récurrence courte. On pose $\mathbf{H}_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$.

L'équivalent de FOM consiste a effectuer m itérations de Lanczos suivies du calcul suivant :

$$\begin{aligned} \mathbf{y}_m &= \|\mathbf{r}_0\|\mathbf{H}_m^{-1}\mathbf{e}_1 \\ \mathbf{x}_m &= \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m \\ \mathbf{r}_m &= \mathbf{b} - \mathbf{A}\mathbf{x}_m = -\beta_{m+1}\mathbf{e}_m^T\mathbf{y}_m\mathbf{v}_{m+1} \end{aligned} \tag{6.14}$$

où l'on observe notamment que la suite des résidus est orthogonale (pour le produit scalaire euclidien).

Algorithm 5: Méthode de Lanczos

Choisir $\mathbf{v}_1$ tel que $\|\mathbf{v}_1\|_2 = 1$, poser $\beta_1 = 0$ et $\mathbf{v}_0 = 0$;
for *j=1,...,m* **do**
  $\tilde{\mathbf{v}}_j = \mathbf{A}\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$;
  $\alpha_j = \tilde{\mathbf{v}}_j^T \mathbf{v}_j$;
  $\tilde{\mathbf{V}}_j \leftarrow \tilde{\mathbf{v}}_j - \alpha_j \mathbf{v}_j$;
  $\beta_{j+1} = \|\tilde{\mathbf{v}}_j\|_2$;
  $\mathbf{v}_{j+1} = \tilde{\mathbf{v}}_j / \beta_{j+1}$;
end

Gradient conjugué

Le gradient conjugué peut être vu comme une adaptation de la méthode FOM dans le cas des matrices symétriques définies positives. Il repose sur l'introduction d'une base $\mathbf{A}$-orthogonale $(\mathbf{w}_j)$ de l'espace de Krylov.

L'idée est la suivante : on factorise $\mathbf{H}_m = \mathbf{L}_m \mathbf{U}_m$ avec

$$\mathbf{L}_m = \begin{pmatrix} 1 & & & \\ \lambda_2 & \ddots & & \\ & \ddots & \ddots & \\ & & \lambda_m & 1 \end{pmatrix} \qquad \mathbf{U}_m = \begin{pmatrix} \delta_1 & \beta_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \beta_m \\ & & & \delta_m \end{pmatrix}$$

On voit que $\mathbf{x}_m = \mathbf{x}_0 + \|\mathbf{r}_0\| \mathbf{V}_m \mathbf{U}_m^{-1} \mathbf{L}_m^{-1} \mathbf{e}_1$ et on note $\mathbf{W}_m = \mathbf{V}_m \mathbf{U}_m^{-1}$ et $\mathbf{z}_m = \|\mathbf{r}_0\| \mathbf{L}_m^{-1} \mathbf{e}_1$. La famille $\mathbf{W}_m$ est $\mathbf{A}$-orthogonale en effet :

$$\mathbf{W}_m^T \mathbf{A} \mathbf{W}_m = \mathbf{U}_m^{-T} \mathbf{V}_m^T \mathbf{A} \mathbf{V}_m \mathbf{U}_m^{-1} = \mathbf{U}_m^{-T} \mathbf{H}_m \mathbf{U}_m^{-1} = \mathbf{U}_m^{-T} \mathbf{L}_m \tag{6.15}$$

cette dernière matrice étant à la fois symétrique et triangulaire inférieure, elle est diagonale.

D'après les calculs précédents, $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{W}_m \mathbf{z}_m$ avec :

$$\begin{aligned} \mathbf{w}_m &= \delta_m^{-1}(\mathbf{v}_m - \beta_m \mathbf{w}_{m-1}) \quad \text{récurrence courte} \\ \lambda_m &= \beta_m / \delta_{m-1} \\ \delta_m &= \alpha_m - \lambda_m \beta_m \end{aligned} \tag{6.16}$$

Si on pose $\mathbf{z}_m = \begin{pmatrix} \mathbf{z}_{m-1} \\ z_m^* \end{pmatrix}$ on voit que $\mathbf{x}_m = \mathbf{x}_{m-1} + z_m^* \mathbf{w}_m$, donc l'approximation se met à jour dans une direction simple, avec au passage $z_m^* = -\lambda_m z_{m-1}^*$.

Le gradient conjugué s'interprète donc donc une méthode de recherche où l'inconnue est mise à jour dans une base $\mathbf{A}$-orthogonale de l'espace de Krylov tel que les résidus soient orthogonaux. Dans l'algorithme suivant, les coefficients α et β ne sont pas les mêmes que ceux de Lanczos.

Algorithm 6: Gradient conjugué

Initialisation $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{w}_0 = \mathbf{r}_0$ **for** *j=1,...,m* **do**
  $\mathbf{p}_j = \mathbf{A}\mathbf{w}_j$;
  $\alpha_j = \frac{\mathbf{r}_j^T \mathbf{r}_j}{\mathbf{p}_j^T \mathbf{r}_j}$;
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j \mathbf{w}_j$;
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j \mathbf{p}_j$;
  $\beta_j = \frac{\mathbf{r}_{j+1}^T \mathbf{r}_{j+1}}{\mathbf{r}_j^T \mathbf{r}_j}$;
  $\mathbf{w}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{w}_j$;
end

Normalement les deux dernières lignes de l'algorithme permettent d'assurer l'$\mathbf{A}$-orthogonalité (la conjugaison) de l'ensemble des $(\mathbf{w}_j)$. Un problème récurent est la perte numérique d'orthogonalité : au bout d'un nombre

important d'itérations, les vecteurs nouvellement générés ne sont plus orthogonaux à ceux générés au début ; les conséquences d'une telle perte d'orthogonalité peuvent être dramatiques sur la convergence des solveurs basés sur ces bases. Diverses stratégies de réorthogonalisation sont possibles. Fondamentalement, on remplace les dernières lignes par $\mathbf{w}_{j+1} = \mathbf{r}_{j+1} - \mathbf{W}_{j+1}(\mathbf{W}_{j+1}^T\mathbf{A}\mathbf{W}_{j+1})^{-1}\mathbf{W}_{j+1}^T\mathbf{A}\mathbf{r}_{j+1}$, ceci dit les mises en oeuvre varient (Gram-Schmidt classique/modifié/itératif).

MinRes

MinRes est la transposition de GMRes dans le cas des matrices symétriques. Il requiert un peu plus de ressources de calcul que le gradient conjugué et est en conséquence moins utilisé.

6.4.3 Algorithmes basés sur la biorthogonalisation de Lanczos

Ces méthodes s'interprètent comme des projections obliques selon des espaces définis à partir de l'espace de Krylov associé à l'opérateur transposé. Elles sont très générales (elles peuvent s'appliquer à des matrices quelconques) mais parce qu'elles ne sont pas associées à des propriétés de minimisation, elles sont difficiles à analyser mathématiquement.

Algorithm 7: Biorthogonalisation de Lanczos

Choisir $\mathbf{v}_1$ et $\mathbf{w}_1$ tels que $\mathbf{v}_1^T\mathbf{w}_1 = 1$;
Poser $\beta_1 = \delta_1 = 0$ et $\mathbf{v}_0 = \mathbf{w}_0 = 0$;
for *j=1,...,m* **do**
  $\alpha_j = \mathbf{w}_j^T\mathbf{A}\mathbf{v}_j$;
  $\hat{\mathbf{v}}_{j+1} = \mathbf{A}\mathbf{v}_j - \alpha_j\mathbf{v}_j - \beta_j\mathbf{v}_{j-1}$;
  $\hat{\mathbf{w}}_{j+1} = \mathbf{A}^T\mathbf{w}_j - \alpha_j\mathbf{w}_j - \delta_j\mathbf{w}_{j-1}$;
  $\delta_{j+1} = |\hat{\mathbf{w}}_{j+1}^T\hat{\mathbf{v}}_{j+1}|^{1/2}$ si $\delta_{j+1} = 0$ stop;
  $\beta_{j+1} = \hat{\mathbf{w}}_{j+1}^T\hat{\mathbf{v}}_{j+1}/\delta_{j+1}$;
  $\mathbf{w}_{j+1} = \hat{\mathbf{w}}_{j+1}/\beta_{j+1}$;
  $\mathbf{v}_{j+1} = \hat{\mathbf{v}}_{j+1}/\delta_{j+1}$;
end

Proposition 6.4.1. *Si l'algorithme n'a pas stoppé avant le rang m alors $\mathbf{V}_m$ et $\mathbf{W}_m$ forment un système biorthogonal :*

$$\mathbf{W}_m^T\mathbf{V}_m = \mathbf{I}_m \tag{6.17}$$

$\mathbf{V}_m$ est une base de $\mathbb{K}(\mathbf{A}, \mathbf{v}_1)$ et $\mathbf{W}_m$ est une base de $\mathbb{K}(\mathbf{A}^T, \mathbf{w}_1)$.

Proposition 6.4.2. *On a les relations suivantes :*

$$\begin{aligned} \mathbf{A}\mathbf{V}_m &= \mathbf{V}_m\mathbf{T}_m + \delta_{m+1}\mathbf{v}_{m+1}\mathbf{e}_m^T \\ \mathbf{A}^T\mathbf{W}_m &= \mathbf{W}_m\mathbf{T}_m^T + \beta_{m+1}\mathbf{w}_{m+1}\mathbf{e}_m^T \\ \mathbf{W}_m^T\mathbf{A}\mathbf{V}_m &= \mathbf{T}_m \end{aligned} \tag{6.18}$$

où $\mathbf{T}_m$ est la matrice tridiagonale suivante :

$$\mathbf{T}_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & & \cdots & & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \tag{6.19}$$

BiCG

Le gradient biconjugué est l'exacte adaptation de la méthode de Lanczos non-symétrique à la résolution de systèmes linéaires. Il permet de fait de résoudre deux systèmes simultanément (associés avec la matrice et la matrice transposée) même si en général un seul est d'intérêt. Sa grosse lacune est l'absence de principe de

Algorithm 8: Gradient biconjugué

Initialisation $\mathbf{x}_0$ et $\tilde{\mathbf{x}}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$ et $\tilde{\mathbf{r}}_0 = \tilde{\mathbf{b}} - \mathbf{A}^T\tilde{\mathbf{x}}_0$, $\mathbf{w}_0 = \mathbf{r}_0$ et $\tilde{\mathbf{w}}_0 = \tilde{\mathbf{r}}_0$;
S'assurer que $\tilde{\mathbf{r}}_0^T\mathbf{r}_0 \neq 0$;
for *j=1,...,m* **do**
  $\mathbf{p}_j = \mathbf{A}\mathbf{w}_j$ et $\tilde{\mathbf{p}}_j = \mathbf{A}^T\tilde{\mathbf{w}}_j$;
  $\alpha_j = \frac{\tilde{\mathbf{r}}_j^T\mathbf{r}_j}{\tilde{\mathbf{p}}_j^T\mathbf{w}_j}$;
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \alpha_j\mathbf{w}_j$ et $\tilde{\mathbf{x}}_{j+1} = \tilde{\mathbf{x}}_j + \alpha_j\tilde{\mathbf{w}}_j$;
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j\mathbf{p}_j$ et $\tilde{\mathbf{r}}_{j+1} = \tilde{\mathbf{r}}_j - \alpha_j\tilde{\mathbf{p}}_j$;
  $\beta_j = \frac{\tilde{\mathbf{r}}_{j+1}^T\mathbf{r}_{j+1}}{\tilde{\mathbf{r}}_j^T\mathbf{r}_j}$;
  $\mathbf{w}_{j+1} = \mathbf{r}_{j+1} + \beta_j\mathbf{w}_j$ et $\tilde{\mathbf{w}}_{j+1} = \tilde{\mathbf{r}}_{j+1} + \beta_j\tilde{\mathbf{w}}_j$;
end

minimisation équivalent, ce qui rend l'étude de sa convergence ardue. Son intérêt est la légèreté des calculs requis contrebalancée par des besoins mémoires importants.

On retrouve (en plus compliqué) les problèmes de perte numérique d'orthogonalité.

6.5 Préconditionnement

Face à un problème mal conditionné (et ils le sont tous), une technique importante est le préconditionnement : l'objectif est de résoudre un système équivalent mieux conditionné que l'original. On distingue :

— Préconditionnement à gauche : $\tilde{\mathbf{A}}^{-1}\mathbf{A}\mathbf{x} = \tilde{\mathbf{A}}^{-1}\mathbf{b}$
— Préconditionnement à droite $\mathbf{A}\tilde{\mathbf{A}}^{-1}\mathbf{y} = \mathbf{b}$ puis $\tilde{\mathbf{A}}^{-1}\mathbf{y} = \mathbf{x}$
— Préconditionnement symétrique $\tilde{\mathbf{L}}^{-1}\mathbf{A}\tilde{\mathbf{L}}^{-T}\mathbf{y} = \tilde{\mathbf{L}}^{-1}\mathbf{b}$ puis $\tilde{\mathbf{L}}^{-T}\mathbf{y} = \mathbf{x}$.

La notation $\tilde{\mathbf{A}}^{-1}$ suggère que le préconditionneur doit autant que possible « ressembler » à l'inverse de la matrice. Le préconditionnement à gauche est très souvent employé. L'intérêt de celui à droite est de ne pas modifier le second membre ce qui permet de mettre à jour le préconditionneur à chaque itération (variante Flexible de GMRes).

Dans le cadre du gradient conjugué, on emploie fréquemment le préconditionnement à gauche ce qui pourrait sembler problématique étant donné qu'il rompt la symétrie de l'opérateur. Cependant si le préconditionneur est symétrique défini positif, on montre que la mise en oeuvre classique (avec préconditionneur à gauche) est équivalente à la résolution par un gradient conjugué avec préconditionneur symétrique (avec $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T = \tilde{\mathbf{A}}$ factorisation de Choleski du préconditionneur). Dans ce cadre, le préconditionnement s'interprète aussi comme une modification de l'orthogonalité entre les espaces.

L'algorithme du GC préconditionné est donné plus bas (avec augmentation).

Il n'existe pas de solution de préconditionnement universel, le plus souvent la connaissance mécanique du système induit un certain préconditionnement, on retient cependant quelques stratégies classiques :

— préconditionneurs diagonaux ou triangulaires par blocs ($\tilde{\mathbf{A}}^{-1} = \mathrm{diag}(\mathbf{A})^{-1}$) ;
— factorisations incomplètes : on résout $\tilde{\mathbf{L}}^{-1}\mathbf{A}\tilde{\mathbf{U}}^{-1}\mathbf{x} = \tilde{\mathbf{L}}^{-1}\mathbf{b}$ où $\tilde{\mathbf{L}}\tilde{\mathbf{U}}$ est une factorisation approchée de la matrice $\mathbf{A}$ qui est par exemple obtenue en ne conservant de la factorisation que les termes dans le profil de $\mathbf{A}$ (dans ce cas les matrices ont la même description du remplissage) ;
— si les calculs sont menés en double précision, on peut envisager de calculer une inverse exacte en single précision ;
— les méthodes de décomposition de domaine algébriques ;
— les méthodes multigrilles.

6.5.1 Méthodes multigrilles

TODO

6.5.2 Méthodes de décomposition de domaine algébriques

TODO

6.5.3 Augmentation

L'augmentation (ou déflation, ou préconditionnement par un projecteur suivant les points de vues) est une technique puissante d'accélération des calculs.

Par souci de simplicité, on l'écrit dans le cas du gradient conjugué, son extension à d'autres méthodes est assez directe. On introduit une matrice $\mathbf{C}$ rectangulaire avec relativement peu de colonnes, que l'on suppose de rang plein. Les colonnes de cette matrice forment donc une base d'un sous-espace (dit d'augmentation). Les méthodes augmentées consistent à modifier le principe de recherche :

$$\begin{aligned} \mathbf{x}_m &\in \mathbf{x}_0 + \mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) \oplus \mathrm{Im}(\mathbf{C}) \\ \mathbf{r}_m &\perp \mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) \oplus \mathrm{Im}(\mathbf{C}) \end{aligned} \tag{6.20}$$

On voit que la méthode revient à imposer une contrainte supplémentaire sur le résidu, vu que cette contrainte est automatiquement vérifiée à convergence (puisque le résidu alors est nul), la contrainte ne perturbe pas la solution, elle peut juste modifier le processus de résolution, et si la matrice $\mathbf{C}$ est bien choisie, l'accélérer.

La mise en œuvre classique repose sur le choix d'une initialisation et une projection adaptées :

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_0 + \mathbf{P}\tilde{\mathbf{x}} \\ \mathbf{C}^T\mathbf{r}_0 = \mathbf{C}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_0) &= 0 \\ \mathbf{C}^T\mathbf{A}\mathbf{P} &= 0 \end{aligned} \tag{6.21}$$

$\mathbf{x}_0$ et $\mathbf{P}$ satisfaisant ces équations (sous-déterminées) ne sont pas uniques, les solutions classiques sont les suivantes :

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{C}\left(\mathbf{C}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{b} \\ \mathbf{P} &= \mathbf{I} - \mathbf{C}\left(\mathbf{C}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{A} \end{aligned} \tag{6.22}$$

L'idée est alors de calculer l'initialisation $\mathbf{x}_0$, d'en déduire le résidu initial $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{P}^T\mathbf{b}$ est d'appliquer le solveur itératif au système $\mathbf{A}\mathbf{P}\tilde{\mathbf{x}} = \mathbf{r}_0$, une fois ce système résolu on en déduit $\mathbf{x} = \mathbf{x}_0 + \mathbf{P}\tilde{\mathbf{x}}$.

Notons au passage que $\mathbf{A}\mathbf{P} = \mathbf{P}^T\mathbf{A} = \mathbf{P}^T\mathbf{A}\mathbf{P}$ et donc le projecteur ne modifie pas la symétrie. De plus, on montre que (dans le cas du gradient conjugué) le système projeté est bien posé est que la solution est unique (après projection finale).

Algorithm 9: Gradient conjugué préconditionné augmenté

Initialisation $\mathbf{x}_0 = \mathbf{C}\left(\mathbf{C}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{b}$, $\mathbf{P} = \mathbf{I} - \mathbf{C}\left(\mathbf{C}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{A}$;
$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\mathbf{z}_0 = \mathbf{M}\mathbf{r}_0$, $\mathbf{w}_0 = \mathbf{P}\mathbf{z}_0$;
for *j=1,...,m* **do**
  $\mathbf{q}_j = \mathbf{A}\mathbf{w}_j$;
  $\alpha_j = (\mathbf{r}_j^T\mathbf{z}_j)/(\mathbf{q}_j^T\mathbf{w}_j)$;
  $\mathbf{x}_{j+1} = \mathbf{x}_j + \mathbf{w}_j\alpha_j$;
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \mathbf{q}_j\alpha_j$;
  $\mathbf{z}_{j+1} = \mathbf{M}\mathbf{r}_{j+1}$;
  $\beta_j = -(\mathbf{q}_{j+1}^T\mathbf{w}_j)/(\mathbf{q}_j^T\mathbf{w}_j)$;
  $\mathbf{w}_{j+1} = \mathbf{P}\mathbf{z}_{j+1} + \mathbf{w}_j\beta_j$;
end

Le choix de la matrice $\mathbf{C}$ est la grande question de l'augmentation. Là encore la mécanique guide le choix. Des applications classiques existent en décomposition de domaine, en multirésolution.

Finalement notons que les techniques d'augmentation sont une des traductions algébriques des concepts de multiéchelle, de multigrille, de réduction de modèle.

Dans le cas d'un solveur en minimum de résidu (GMRes, orthomin,...), il faut adapter la contrainte :

$$\begin{aligned} \mathbf{x}_m &\in \mathbf{x}_0 + \mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) \oplus \mathrm{Im}(\mathbf{C}) \\ \mathbf{r}_m &\perp \mathbf{A}\left(\mathbb{K}_m(\mathbf{A}, \mathbf{r}_0) \oplus \mathrm{Im}(\mathbf{C})\right) \end{aligned} \tag{6.23}$$

La mise en oeuvre classique repose sur le choix d'une initialisation et une projection adaptées :

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_0 + \mathbf{P}\tilde{\mathbf{x}} \\ \mathbf{C}^T\mathbf{A}^T\mathbf{r}_0 = \mathbf{C}^T\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_0) &= 0 \\ \mathbf{C}^T\mathbf{A}^T\mathbf{A}\mathbf{P} &= 0 \end{aligned} \tag{6.24}$$

$\mathbf{x}_0$ et $\mathbf{P}$ satisfaisant ces équations (sous-déterminées) ne sont pas uniques, les solutions classiques sont les suivantes :

$$\begin{aligned} \mathbf{x}_0 &= \mathbf{C}\left(\mathbf{C}^T\mathbf{A}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{A}^T\mathbf{b} \\ \mathbf{P} &= \mathbf{I} - \mathbf{C}\left(\mathbf{C}^T\mathbf{A}^T\mathbf{A}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{A}\mathbf{A}^T \end{aligned} \tag{6.25}$$

6.6 Solveurs à membres de droite multiples

Les solveurs de Krylov peuvent résoudre simultanément plusieurs systèmes linéaires, et les résolutions simultanées bénéficient les unes des autres. Le résultat principal est que si on résout deux systèmes simultanément, la résolution se fait en au plus le max du nombre d'itérations pour résoudre chacun des systèmes. En plus on exploite des méthodes blocs pour les produits, ce qui est avantageux.

L'idée est de générer à chaque itération autant de directions de recherche que de seconds membres, et d'optimiser chaque approximation sur le sous-espace engendré. Le principal problème avec ces méthodes est le risque de se retrouver avec des directions colinéaires, ce qui se traduit conduit à devoir prendre quelques précautions numériques. On donne un gradient conjugué à plusieurs seconds membres, la pseudo-inversion est requise quand les directions deviennent colinéaires.

Algorithm 10: Gradient conjugué pour N membres de droite

On note en majuscule les groupes de vecteurs (matrices $n \times N$);
Initialisation $\mathbf{X}_0$, $\mathbf{R}_0 = \mathbf{B} - \mathbf{A}\mathbf{X}_0$, $\mathbf{W}_0 = \mathbf{R}_0$;
for *j=1,...,m* **do**
  $\mathbf{P}_j = \mathbf{A}\mathbf{W}_j$;
  $\alpha_j = (\mathbf{P}_j^T\mathbf{W}_j)^+(\mathbf{R}_j^T\mathbf{R}_j)$, matrice $N \times N$;
  $\mathbf{X}_{j+1} = \mathbf{X}_j + \mathbf{W}_j\alpha_j$;
  $\mathbf{R}_{j+1} = \mathbf{R}_j - \mathbf{P}_j\alpha_j$;
  $\beta_j = -(\mathbf{P}_j^T\mathbf{W}_j)^+(\mathbf{P}_{j+1}^T\mathbf{W}_j)$, matrice $N \times N$;
  $\mathbf{W}_{j+1} = \mathbf{R}_{j+1} + \mathbf{W}_j\beta_j$;
end

Chapitre 7

Problèmes aux valeurs propres

On cherche des couples $(\lambda, \mathbf{u})$ solutions de $\mathbf{Au} = \lambda\mathbf{u}$ (dans $\mathbb{R}^n$). Pour des matrices de taille supérieure à 4, il n'existe pas de méthode directe pour les obtenir. Si $\mathbf{U}$ est la matrice des vecteurs propres et $\mathbf{\Lambda}$ la matrice diagonale des valeurs propres, on a : $\mathbf{AU} = \mathbf{U\Lambda}$. On ordonne les valeurs propres $\lambda_1 \geqslant \lambda_2 \dots \lambda_n$.

On rappelle au passage les problèmes aux valeurs propres généralisées $\mathbf{Au} = \lambda\mathbf{Bu}$ qui se traitent pratiquement de la même manière.

7.1 Techniques de base

7.1.1 La méthode des puissances

Algorithm 11: Méthode des puissances

Choisir $\mathbf{v}_0$;
for *j=1,...,m* **do**
  $\mathbf{v}_j = \mathbf{A}\mathbf{v}_{j-1}$;
  $\alpha_j = \|\mathbf{v}_j\|$;
  $\mathbf{v}_j \leftarrow \mathbf{v}_j/\alpha_j$;
end

Si la plus grande valeur propre dont le sous-espace propre rencontre $\mathbf{v}_0$ est semi-simple alors (α_j) converge vers celle-ci et $\mathbf{v}_j$ converge vers un vecteur propre associé.

Remarque. Si $\mathbf{v}_0$ est choisi au hasard alors la probabilité qu'il rencontre $\mathbf{u}_1$ est de 1. Autrement dit $\mathbf{v}_0 = \sum v_i\mathbf{u}_i$ et $v_1 \neq 0$ presque sûrement.

La convergence est d'autant plus rapide que cette valeur propre est séparée de la seconde plus grande valeur propre. En effet, on a décompose $\mathbf{v_0}$ sur la base propre $\mathbf{v}_0 = \sum v_k\mathbf{u}_k$, on a :

$$\begin{aligned} \mathbf{A}^m\mathbf{v}_0 &= \sum_k \lambda_k^m v_k \mathbf{u}_k \\ &= \lambda_1^m v_1 \left(\mathbf{u}_1 + \sum_{k>1} \left(\frac{\lambda_k}{\lambda_1}\right)^m \frac{v_k}{v_1}\mathbf{u}_k \right) \end{aligned} \tag{7.1}$$

On a $\|\mathbf{A}^m\mathbf{v}_0\| \approx \lambda_1^m v_1$, donc $\mathbf{A}^m\mathbf{v}_0/\|\mathbf{A}^m\mathbf{v}_0\| \to \mathbf{u}_1$ et que le terme restant qui tend le moins vite vers 0 est de la forme $(\lambda_2/\lambda_1)^m$.

Remarque (Méthode à pas multiple). Tant que les calculs n'explosent pas, on peut se passer de l'étape de normalisation dans les itérations. Au lieu de calculer $\mathbf{Av_j}$, on calcule $\mathbf{A}^m\mathbf{v_j}$. Dans certains cas (matrice stockée de manière distribuée), cela permet de gagner du temps de calcul.

Algorithm 12: Méthode des puissances inverses

Choisir $\mathbf{v}_0$;
for *j=1,...,m* **do**
  $\mathbf{v}_j = \mathbf{A}^{-1}\mathbf{v}_{j-1}$;
  $\alpha_j = \|\mathbf{v}_j\|_2$;
  $\mathbf{v}_j \leftarrow \mathbf{v}_j/\alpha_j$;
end

7.1.2 Méthode des puissances inverses

On suppose que la matrice $\mathbf{A}$ est inversible. Cette méthode converge vers un couple (vecteur, valeur) propre associé à la plus petite valeur propre de $\mathbf{A}$. La convergence est d'autant plus rapide que la seconde plus petite valeur propre est loin de la première.

7.1.3 Méthode des puissances inverses avec décalage

Pour étudier l'intérieur du spectre d'une matrice, on décale son origine. Soit σ une approximation de la valeur propre à trouver. On s'en sert de décalage (*shift*) : $(\mathbf{A} - \sigma\mathbf{I})$.

Algorithm 13: Méthode des puissances inverses avec décalage

Choisir $\mathbf{v}_0$;
for *j=1,...,m* **do**
  $\mathbf{v}_j = (\mathbf{A} - \sigma\mathbf{I})^{-1}\mathbf{v}_{j-1}$;
  $\alpha_j = \|\mathbf{v}_j\|$;
  $\mathbf{v}_j \leftarrow \mathbf{v}_j/\alpha_j$;
end
Valeur après recalage : $\alpha_m + \sigma$.

Si le décalage est proche de la valeur recherchée (et donc loin de la seconde plus proche valeur) alors la convergence sera très rapide.

Une idée très intéressante dans le cas Hermitien est d'optimiser le décalage à chaque itération en utilisant le quotient de Rayleigh pour approcher la valeur propre recherchée.

Algorithm 14: Itération sur le quotient de Rayleigh

Choisir $\mathbf{v}_0$ tel que $\|\mathbf{v}_0\|_2 = 1$;
for *j=1,...,m* **do**
  $\sigma_j = \mathbf{v}_{j-1}^T\mathbf{A}\mathbf{v}_{j-1}$;
  $\mathbf{v}_j = (\mathbf{A} - \sigma_j\mathbf{I})^{-1}\mathbf{v}_{j-1}$;
  $\alpha_j = \|\mathbf{v}_j\|_2$;
  $\mathbf{v}_j \leftarrow \mathbf{v}_j/\alpha_j$;
end

7.1.4 Déflation

Si on a déjà calculé une valeur propre λ_j et un vecteur propre (à droite) associé $\mathbf{u}_j$, et que l'on cherche la seconde plus proche, une solution consiste à modifier l'opérateur pour faire disparaître cette valeur propre, une solution est la déflation de Wielandt :

$$\mathbf{A}_1 = \mathbf{A} - \sigma\mathbf{u}_j\mathbf{v}^H \tag{7.2}$$

σ est un décalage à déterminer, $\mathbf{v}$ doit juste satisfaire $\mathbf{v}^H\mathbf{u}_j = 1$. Cette déflation ne modifie pas le spectre sauf pour λ qui est décalé vers $\lambda_j - \sigma$. Le choix du vecteur $\mathbf{v}$ peut se faire selon plusieurs critères. En pratique le choix $\mathbf{v} = \mathbf{u}_j$ se révèle quasi-optimal (dans l'optique de minimiser le conditionnement du problème de recherche de λ_{j+1}) pour un *shift* suffisamment grand ; un autre choix intéressant est d'utiliser $\mathbf{v} = \mathbf{w}_j$ où $\mathbf{w}_j$ est le vecteur

propre à gauche associé à λ_j (déflation d'Hotelling), bien sûr les deux stratégies sont équivalentes dans le cas hermitien.

Il est possible de faire de la déflation par sous-espaces $\mathbf{A}_j = \mathbf{U}_j \sigma \mathbf{U}_j^H$ où $\mathbf{U}_j$ est une base orthonormale de vecteurs de Schur (vecteurs intervenant dans la trigonalisation éponyme). On peut alors enchaîner les calculs de valeurs propres en augmentant la base de déflation par l'orthonormalisé d'un vecteur convergé.

7.1.5 Itérations par sous-espaces

Dans toutes les méthodes précédentes, il est possible de rechercher simultanément un ensemble de (valeurs, vecteurs) propres. On parle de méthode par sous-espace ou par bloc. On donne la version puissance itérée. L'idée est de partir d'une famille initiale de m vecteurs orthonormaux $\mathbf{X}_0 = (\mathbf{x}_{0,0}, \ldots, \mathbf{x}_{m,0})$, il suffit simplement de remplacer la normalisation par une factorisation QR.

Algorithm 15: Itération par sous-espaces simple

Choisir m vecteurs initiaux orthonormaux $\mathbf{X}_0$;
for $j = 1, \cdots$ *convergence* **do**
  Calculer $\mathbf{X}_j = \mathbf{A}\mathbf{X}_{j-1}$;
  Calculer la factorisation $\mathbf{X}_j = \mathbf{Q}\mathbf{R}$ et poser $\mathbf{X}_j = \mathbf{Q}$;
end

Ces méthodes permettent de converger vers des vecteurs de Schur (donc des sous-espaces stables). On peut y appliquer toutes les variations précédentes (déflation, shift, pas multiples). En particulier, les pas multiples permettent d'économiser des factorisations QR.

7.2 Méthodes de projection

Les méthodes de projection consistent à chercher les valeurs propres sur des sous-espaces :

$$\begin{aligned} \tilde{\mathbf{u}} &\in \mathbb{V} \\ \mathbf{A}\tilde{\mathbf{u}} - \tilde{\lambda}\tilde{\mathbf{v}} &\perp \mathbb{W} \end{aligned} \tag{7.3}$$

On retrouve les conditions de Petrov-Galerkin. Le problème se ramène à :

$$\begin{aligned} \tilde{\mathbf{u}} &= \mathbf{V}\mathbf{y} \\ \left(\mathbf{W}^H\mathbf{A}\mathbf{V}\right)\mathbf{y} &= \tilde{\lambda}\left(\mathbf{W}^H\mathbf{V}\right)\mathbf{y} \end{aligned} \tag{7.4}$$

qui est un problème aux valeurs propres classique si on peut choisir des bases biorthonormales $\left(\mathbf{W}^H\mathbf{V}\right) = \mathbf{I}$, ce qui est possible si aucun vecteur de $\mathbb{W}$ n'est orthogonal à $\mathbb{V}$. Typiquement $\mathbb{W} = \mathbf{B}\mathbb{V}$ et il suffit que $\mathbf{B}$ soit bien fait (par exemple, $\mathbf{B}$ SPD). Pour la suite on se place donc dans le cas $\left(\mathbf{W}^H\mathbf{V}\right) = \mathbf{I}$. Si $\left(\tilde{\lambda}, \mathbf{y}\right)$ est solution du problème au valeur propre précédent, et si $\tilde{\mathbf{u}} = \mathbf{V}\mathbf{y}$, on a

$$\mathbf{V}\left(\mathbf{W}^H\mathbf{A}\mathbf{V}\right)\mathbf{W}^H\tilde{\mathbf{u}} = \tilde{\lambda}\tilde{\mathbf{u}} \tag{7.5}$$

La matrice $\mathbf{V}\left(\mathbf{W}^H\mathbf{A}\mathbf{V}\right)\mathbf{W}^H$ est la projection de $\mathbf{A}$ sur $\mathbb{V}$ orthogonalement à $\mathbb{W}$.

Quand on choisit pour espaces les sous-espaces de Krylov, la matrice projetée est Hessenberg (ou tridiagonale pour les méthodes de Lanczos) et le calcul des valeurs propres est simplifié.

Dans le cas d'une projection orthogonale (Galerkin), on a l'algorithme de Rayleigh-Ritz pour le calcul des valeurs propres. Dans le cas Hermitien les éléments propres obtenus par la procédure de Rayleigh-Ritz sont

Algorithm 16: Procédure de Rayleigh-Ritz

Calculer une base orthonormale $\mathbf{V}_m$ de $\mathbb{V}$;
Calculer $\mathbf{A'}_m = \mathbf{V}_m^H\mathbf{A}\mathbf{V}_m$;
Calculer les valeurs propres $(\tilde{\lambda}_i)$ de $\mathbf{A'}_m$, sélectionner les valeurs souhaitées;
Calculer les vecteurs propres $(\mathbf{y}_i)$ de $\mathbf{A'}_m$ souhaités, poser $\tilde{\mathbf{u}}_i = \mathbf{V}_m\mathbf{y}_i$;

optimaux dans un certain sens.

Dans la cas d'une projection en minimum de résidu ($\mathbf{W} = \mathbf{AV}$), on parle des vecteurs Ritz harmoniques, le problème aux valeurs propres projeté s'écrit :

$$\begin{aligned} \tilde{\mathbf{u}} &= \mathbf{V}\mathbf{y} \\ \left(\mathbf{V}^T\mathbf{A}^T\mathbf{A}\mathbf{V}\right)\mathbf{y} &= \tilde{\lambda}\left(\mathbf{V}^T\mathbf{A}^T\mathbf{V}\right)\mathbf{y} \\ \left(\mathbf{V}^T\mathbf{A}^T\mathbf{A}\mathbf{V}\right)\mathbf{y} &= \tilde{\lambda}\left(\mathbf{V}^T\mathbf{A}^T\mathbf{A}^{-1}\mathbf{A}\mathbf{V}\right)\mathbf{y} \\ &\text{donc} \end{aligned} \tag{7.6}$$

on reconnaît une méthode de Rayleigh-Ritz sur la matrice inverse projetée dans le sous-espace $\mathbf{AV}$.