Fortran... et puis quoi encore ?

Vincent Magnin, José Alves, Antoine Arnoud, Arjen Markus, Michele Esposito Marzino

Fortran... et puis quoi encore ?

Vincent Magnin [1], José Alves [2], Antoine Arnoud [3], Arjen Markus [4], Michele Esposito Marzino [5]

Le Fortran moderne est un langage normalisé incluant les paradigmes de la programmation orientée objet et de la programmation parallèle. La communauté Fortran-lang, créée fin 2019, travaille activement à la modernisation de son environnement de développement. De nouveaux compilateurs sont en développement. Et la quatrième norme Fortran du XXI^{e} *siècle devrait être publiée à l'automne 2023.*

Genèse et période classique

Le premier compilateur optimiseur

Master de mathématiques en poche, le jeune John Backus est embauché chez IBM en 1950 pour travailler sur les calculateurs de l'entreprise. Pour alléger son travail de programmation en langage machine, il conçoit d'abord un interpréteur nommé Speedcoding pour l'IBM 701, avec en particulier une émulation du calcul en virgule flottante. Puis en décembre 1953, il propose à son supérieur le projet FORTRAN [6] *(mathematical FORmula TRANslating system).* La programmation en

1. Maître de conférences, IEMN (UMR CNRS 8520), vincent.magnin@univ-lille.fr.

2. Responsable d'équipe de développement en multiphysique et deep learning, Transvalor S.A., Sophia Antipolis, jose.alves@transvalor.com.

3. Économiste, *International Monetary Fund*, antoine.arnoud@gmail.com.

4. Responsable de modèles numériques dédiés à la qualité de l'eau et l'écologie, *Deltares Research Institute*, Pays-Bas, arjen.markus895@gmail.com.

5. Doctorant à l'université de Liège, michele.espositomarzino@uliege.be.

6. Le nom sera choisi fin 1954, la proposition initiale parlant d'un « système de codage automatique ». Notons qu'on écrit FORTRAN pour la période antérieure à Fortran 90.

DOI:10.48556/SIF.1024.22.143 **1024** – Bulletin de la SIF, numéro 22, novembre 2023, p. 143–161

langage machine exige des experts et le coût d'un calcul scientifique est au moins pour moitié dû à leurs salaires (programmation, tests, *debugging*). Un langage de haut niveau permettrait de diminuer drastiquement le temps de travail nécessaire. Cet argument économique permet à Backus d'obtenir rapidement une équipe pour travailler sur un langage et son compilateur [18]. Tout reste à inventer : si l'idée d'un langage compilé est déjà dans l'air avec par exemple les travaux de Grace Hopper sur le langage A-0, le compilateur [7] FORTRAN est considéré comme le premier compilateur optimiseur. L'équipe est en effet consciente que pour qu'il soit accepté, un code optimal doit être généré quelle que soit l'application. A l'époque, rares sont ceux prêts à croire qu'une machine puisse générer un code machine aussi rapide que celui écrit par un être humain. Et Von Neumann lui-même ne voit pas l'intérêt d'un langage de haut niveau [28].

La machine ciblée est le futur IBM 704 et Backus insiste pour qu'elle devienne la première machine travaillant directement avec des réels à virgule flottante [8]. Après trois ans de travail, le compilateur est enfin déployé en avril 1957. C'est une révolution. Mary Tsingou, physicienne et mathématicienne au *Los Alamos National Laboratory*, décrit la situation : « *Quand le Fortran est arrivé, c'était presque comme le paradis* » [23]. Sa syntaxe est en effet simple et adaptée aux calculs scientifiques comme le signifie son nom. Le manuel de référence du programmeur, prêt dès octobre 1956, tient en 54 pages [7] et les bases du langage peuvent donc être apprises en quelques heures. Frances Elizabeth Allen, embauchée à 24 ans en juillet 1957, est chargée d'enseigner le FORTRAN aux scientifiques d'IBM et le plus difficile est de les convaincre de son utilité et efficacité. Mais lire les 20 000 lignes de code du compilateur orientera sa carrière. Elle recevra en effet le prix Turing 2006 « *pour ses contributions pionnières à la théorie et à la pratique des techniques utilisés par les compilateurs optimiseurs qui ont jeté les bases des compilateurs optimiseurs modernes et de l'exécution parallèle automatique* [16]. »

Conçu avec uniquement en tête l'IBM 704, le compilateur FORTRAN s'avère rapidement adapté à d'autres machines en commençant par celles de la marque, faisant du langage le premier langage de programmation portable. Le programmeur peut désormais s'abstraire du langage machine et le scientifique s'improviser programmeur [28]. Rien d'étonnant donc à ce que ce compilateur fasse partie, d'après un article du journal *Nature* publié en 2021, des dix codes informatiques qui ont le plus transformé la science [34].

Le premier langage normalisé

Revers de la médaille, une normalisation est rapidement rendue nécessaire pour conserver la portabilité du langage, chaque éditeur de compilateur y ajoutant ses

7. En fait, les articles de l'époque utilisent le mot *translator*, plutôt que *compiler.*

8. Dans les PC, il faudra attendre 1989 avec le microprocesseur 32 bits Intel i486 pour que le coprocesseur arithmétique à virgule flottante (FPU) ne soit plus optionnel.

extensions. Il devient donc le premier langage à être normalisé avec la norme FORTRAN 66 (ANSI X3.9-1966) qui comporte deux versions du langage : le langage complet et une version simplifiée nommée Basic FORTRAN. Une version intermédiaire ECMA-9 est même normalisée un an plus tôt par l'*European Computer Manufacturers Association* (ECMA).

FORTRAN 77

La norme FORTRAN 77 (ANSI X3.9-1978), complétée par l'extension MIL-STD-1753 du département de la Défense américain, est une évolution majeure qui facilite en particulier la programmation structurée.

L'extension militaire introduit par ailleurs l'instruction `IMPLICIT NONE`, que l'on utilise désormais systématiquement, qui permet de désactiver le typage implicite des variables. En effet, par défaut les variables dont le nom commence par I, J, K, L, M, ou N sont de type entier, les autres de type réel. Voilà donc les classiques compteurs de boucles `i, j, k` que nous utilisons toujours !

Le langage restera dominant jusqu'à la fin des années 1970. Au point même que de nombreux artistes l'ont utilisé pour explorer les nouvelles possibilités offertes par l'ordinateur : par exemple pour faire de la musique algorithmique (Iannis Xenakis et Pierre Barbaud), des œuvres graphiques (Hiroshi Kawano), des générateurs de poèmes, etc. Et FORTRAN 77 restera une référence pour longtemps, trop longtemps !

Un prix Turing

Entre autres distinctions, John Backus reçoit le prix Turing 1977 « *pour ses contributions profondes, influentes et durables à la conception de systèmes de programmation pratiques de haut niveau, notamment par ses travaux sur le FORTRAN, et pour ses publications pionnières sur les procédures formelles pour la spécification des langages de programmation* [19]. » Le projet FORTRAN mené à terme, il est en effet parti rapidement vers d'autres aventures. Sa participation à ALGOL 58 et 60 l'amène à travailler sur ce que l'on appellera la forme de Backus-Naur (BNF), notation formelle permettant de décrire un langage de programmation. Nommé *IBM Fellow* en 1963, il travaillera ensuite à sa vision de ce que pourrait être un style de programmation fonctionnelle, sa lecture Turing s'intitulant *Can programming be liberated from the Von Neumann style ? : a functional style and its algebra of programs* [9].

9. Peut-on libérer la programmation du style Von Neumann ? Un style fonctionnel et son algèbre des programmes.

Fortran moderne

L'expression *Fortran moderne*, souvent utilisée dans les titres des livres anglo-saxons depuis une vingtaine d'années [20, 31, 32], n'a rien d'officiel : certains l'utiliseront à partir de Fortran 90/95, d'autres à partir de Fortran 2003. Elle pourrait être perçue comme un oxymore, figure de style courante à notre époque, mais il n'en est rien !

Fortran 90

La norme Fortran 90, arrivée avec plusieurs années de retard, est une version majeure comme le suggère le changement de casse : on n'écrit plus FORTRAN mais Fortran. Il s'agit également de sa première norme ISO, gérée par le comité international ISO/IEC JTC1/SC22/WG5 et le comité technique américain X3J3. Parmi les nombreuses nouveautés, on note le format de code source *free form* qui permet enfin d'écrire du code sans se soucier du rôle de certaines colonnes hérité des cartes perforées (format *fixed form*, généralement associé à l'extension `.f`). On notera que l'extension la plus utilisée de fait pour le format libre est `.f90`, ce qui peut prêter à confusion pour le débutant, le code pouvant être conforme à n'importe quelle norme ultérieure !

Plutôt que de définir de nouveaux types, on introduit le concept de *kind* pour définir des variantes potentielles de tous les types de base (`real`, `complex`, `integer`, `logical`, `character`). Pour les `integer` et `real`, des fonctions intrinsèques permettent de sélectionner une variante, si elle est supportée par le compilateur, en fonction des caractéristiques numériques souhaitées.

Le découpage du code source en modules permet un embryon de programmation orientée objet, chaque module embarquant ses propres données et routines, et facilite les diagnostics des compilateurs. Les attributs `public` (par défaut) et `private` permettent d'indiquer si une entité sera accessible ou non en-dehors du module. De plus, le concept de type dérivé permet à l'utilisateur de définir ses propres structures de données :

```
type personne
   character(30) :: prenom
   integer :: age
   real :: poids
end type

type(personne) :: lui = personne("Jean", 50, 70.5)
print *, "Son prenom est : ", lui%prenom
```

On notera l'utilisation du caractère `%` au lieu du classique point pour accéder à un champ de la structure. L'introduction des types dérivés est d'autant plus importante

qu'ils serviront de base à l'introduction de la programmation orientée objet dans la norme Fortran 2003.

Une syntaxe orientée tableau permet de simplifier le code en évitant de nombreuses boucles. On définit par exemple ci-dessous trois vecteurs à trois composantes [10], on en calcule et affiche la somme puis l'on calcule un vecteur dont les composantes sont les racines carrées des composantes du premier. Fortran 90 introduit en effet les fonctions et procédures dites `elemental` qui peuvent s'appliquer aussi bien à un scalaire qu'à un tableau.

```
program tableaux
   implicit none
   real :: u(3) = [ 1.0, 2.0, 3.0 ]
   real :: v(3) = [ 4.0, 5.0, 6.0 ]
   real :: w(3)

   w = u + v
   print *, "Somme des vecteurs u et v : ", w
   print *, "Vecteur racines carrees des u(i) : ", sqrt(u)
end program tableaux
```

De plus, les tableaux peuvent désormais être alloués dynamiquement. Ils sont dans ce cas déclarés avec l'attribut `allocatable` et leurs dimensions sont alors définies plus tard à l'aide de l'instruction `allocate()`. On peut libérer l'espace occupé avec l'instruction `deallocate()`.

Les pointeurs (attribut `pointer`) font leur apparition en Fortran, ce qui facilite entre autres l'utilisation de listes, d'arbres, de graphes, etc. Mais alors qu'en langage C un pointeur est une adresse mémoire, le concept de pointeur Fortran s'apparente plutôt à un alias. Une entité ne peut par ailleurs être pointée que si elle a été déclarée avec l'attribut `target` ou si elle est elle-même un pointeur. Ces propriétés permettent aux compilateurs de générer un code robuste et optimisé.

Enfin, deux nouvelles sections apparaissent dans les normes Fortran : la liste des caractéristiques obsolètes et celle des caractéristiques supprimées. Dans la pratique, ces dernières restent généralement supportées par les compilateurs afin que les codes anciens continuent de fonctionner.

Fortran 95

Il s'agit d'une norme mineure qui vient compléter la précédente. Fortran 90 avait introduit la possibilité d'allouer de la mémoire et d'utiliser des pointeurs vers les tableaux. Mais ces fonctionnalités avaient leurs limites, le programmeur devant par exemple gérer la mémoire de façon explicite. Fortran 95 offre beaucoup plus de

10. En toute rigueur, la syntaxe pour initialiser un tableau en Fortran 90 est `(/ /)`, mais la norme Fortran 2003 introduit la syntaxe entre crochets, plus lisible et intuitive.

liberté et de flexibilité : la désallocation automatique libère le programmeur d'une grande partie de la gestion de la mémoire, ce qui limite les problèmes de fuite de mémoire et lui permet de se concentrer encore plus sur ce qu'il veut faire plutôt que sur les détails techniques pour y parvenir.

Une fonction peut recevoir l'attribut `pure` indiquant qu'elle ne produit pas d'effet de bord (modification d'une variable globale, d'un argument, réalisation d'entrées/-sorties, etc.) et peut donc être utilisée sans problème dans un calcul parallèle. Le programmeur peut également définir ses propres fonctions `elemental`, forme particulière de fonction `pure`.

Notons en passant qu'une des méthodes utilisées pour faire évoluer le langage est donc d'ajouter des attributs tels que `elemental`, `pure`, `intent`, `allocatable`, `pointer`, `target`... Ils assurent de plus la robustesse du code, le compilateur étant à chaque fois amené à effectuer les vérifications nécessaires.

Enfin, première tentative d'introduire dans le langage lui-même le calcul parallèle, les boucles `forall` seront finalement déclarées obsolètes dans la norme Fortran 2018, leur complexité ayant rendu difficile une implémentation efficace par les compilateurs.

Fortran 2003

Tout comme Fortran 90, la norme Fortran 2003 est une version majeure, comme en témoignent ses 585 pages, soit 55 % de plus que la norme précédente. Elle apporte en particulier la programmation orientée objet (POO) [36], en se basant sur les modules et les types dérivés introduits par Fortran 90. Une classe est ainsi un type dérivé contenant à la fois des données et des méthodes, appelées procédures liées au type *(type-bound procedures)*, comme dans cet exemple très simple :

```
type :: vecteur2D
   real :: x, y
   contains
   procedure :: ajouter => ajouter_vecteur2D
   procedure :: rotation => rotation_vecteur2D
end type vecteur2D
```

L'introduction des classes rend possible une véritable POO : plus ou moins inspirée du concept de classe dans des langages comme C++ et Java, une classe Fortran peut être abstraite (attribut `abstract`) ou non et peut être étendue (attribut `extends`). Mais on peut également personnaliser les méthodes des objets appartenant à une même classe en utilisant des pointeurs de procédure, la syntaxe étant similaire, et aboutir ainsi à un style de programmation orientée prototype [12].

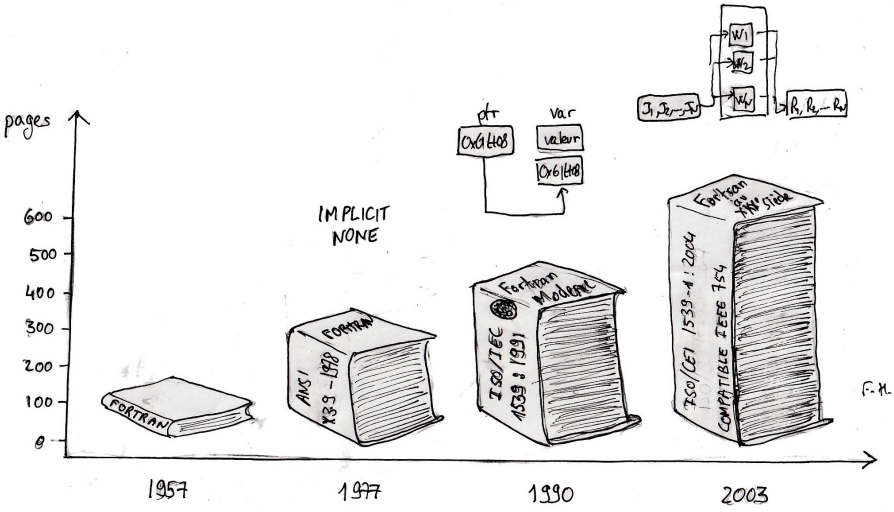
Pour faciliter encore plus la construction de types de données abstraits tels que des listes chaînées, on peut également utiliser ce que l'on appelle des variables polymorphes illimitées. On pourrait les qualifier de caméléonesques car elles prennent la

valeur qui leur est attribuée, tout en conservant le type initial et l'on conserve ainsi un typage sûr [31].

Autre apport important, l'interopérabilité avec le langage C est enfin normalisée. Il devient aisé d'appeler des bibliothèques C d'une façon portable, par exemple pour créer des interfaces graphiques avec GTK [29] ou gérer des bases de données avec SQLite [31]. Réciproquement, cela facilite l'écriture de bibliothèques Fortran pour d'autres langages, par exemple Python, les faisant ainsi bénéficier de sa vitesse de calcul et de ses possibilités en termes de précision numérique. Dans la pratique, le nouveau module intrinsèque `iso_c_binding` contient des constantes permettant de faire la correspondance entre les types de données des deux langages, ainsi que des fonctions facilitant la correspondance entre leurs pointeurs. L'ordre de stockage en mémoire des tableaux multidimensionnels étant différent dans les deux langages, le programmeur doit toutefois veiller à inverser l'ordre des indices lorsqu'un tel tableau est passé d'un langage à l'autre.

La procédure `move_alloc()` et l'allocation implicite lors de l'affection simplifient toutes deux la gestion dynamique des objets `allocatable`, notamment la réallocation.

Parmi les nombreuses autres nouveautés, on notera les améliorations des entrées-sorties (transferts asynchrones par exemple), trois modules intrinsèques permettant d'accéder aux fonctionnalités de la norme IEEE 754 (on peut par exemple modifier le mode d'arrondis du microprocesseur), le support optionnel des caractères Unicode (ISO/IEC 1064619 1 :2000 UCS-4), l'accès aux arguments de la ligne de commande et aux variables d'environnement.



Calcul parallèle

Certains codes Fortran sont destinés à tourner sur ordinateur personnel, d'autres sont destinés à être lancés sur des supercalculateurs. Il faut donc gérer le parallélisme à différentes échelles et sur différents types d'architectures [30] : microprocesseurs multi-coeurs, vectorisation [11], processeurs graphiques (GPU), machines multi-processeurs d'architectures diverses, machines en réseaux, etc. Mettre en place des standards de calcul parallèle et les implémenter de façon efficace est donc une tâche difficile.

Les premières tentatives remontent à la deuxième moitié des années 80 au moment où d'une part les supercalculateurs évoluent d'un modèle vectoriel vers un modèle multi-processeurs et où d'autre part le fonctionnement en réseau commence à se généraliser. Ainsi en 1989, PVM *(Parallel Virtual Machine)* [13] est une des premières solutions pour faire du calcul parallèle en C ou Fortran sur un réseau de machines hétérogènes (Unix/Windows).

Fin 1991, le travail pour normaliser un *High Performance Fortran (HPF)* commence [27]. Tout restant à inventer, cette version du langage n'arrivera finalement pas à convaincre et n'aura pas le temps d'arriver à maturité, sauf au Japon qui développera son dialecte HPF/JA et l'utilisera sur le supercalculateur Earth Simulator, le plus puissant au monde de 2002 à 2004 [6]. Mais HPF aura permis de défricher le terrain et un certain nombre de graines seront semées. Par exemple, l'ajout de directives au compilateur en commentaires afin que le programme reste compilable sur des machines non parallèles se retrouvera dans OpenMP ou OpenACC, les boucles `forall` et l'attribut `pure` dans Fortran 95.

L'inclusion des paradigmes du calcul parallèle dans les normes Fortran elles-mêmes va nécessiter énormément de temps et d'expérimentations. Il faudra attendre pour cela Fortran 2008 et 2018, avec l'effort considérable que cela impliquera au niveau du développement des compilateurs.

Standards externes

Deux standards externes émergent dans les années 90 pour les langages C, C++ et Fortran [27]. Sorti en 1994, MPI *(Message Passing Interface)* permet la programmation parallèle par passage de messages entre les processus. Il a été conçu pour fonctionner aussi bien avec des systèmes à mémoire partagée qu'avec des systèmes à mémoire distribuée. Dans ce dernier cas, il peut s'agir d'ordinateurs en réseau, de noeuds d'un cluster de calcul, etc. Puissant mais exigeant de l'utilisateur qu'il contrôle finement les communications entre processus, MPI est parfois surnommé

11. Instructions SIMD (*single instruction, multiple data*) telles que FMA, SSE, AVX, etc. Les compilateurs modernes sont capables de vectoriser certaines boucles automatiquement et des standards tels que OpenMP comportent des directives SIMD permettant au développeur de guider le compilateur dans les cas complexes.

l'assembleur du calcul parallèle. Apparu en 1997, OpenMP *(Open Multi-Processing)* est quant à lui destiné aux systèmes à mémoire partagée. Il permet à l'aide de directives en commentaires de paralléliser assez facilement un code existant, si les algorithmes s'y prêtent.

La montée en puissance des GPU dans les années 2000 va faire émerger différents systèmes permettant de déporter les calculs vers les nombreux coeurs de ces processeurs, avec en particulier le CUDA *(Compute Unified Device Architecture)* de NVIDIA en 2007, OpenCL 1.0 *(Open Computing Language)* en 2009, OpenACC 2.0 *(Open Accelerators)* en 2012.

Mais ces standards sont extérieurs au Fortran et les deux normes suivantes vont introduire les paradigmes du calcul parallèle dans le langage lui-même.

Fortran 2008

D'abord développés comme une extension de Fortran 95, les co-tableaux *(coarrays)* entrent dans la norme Fortran 2008. Plusieurs copies d'un programme peuvent être exécutées en parallèle, que ce soit sur un microprocesseur multi-coeurs ou un supercalculateur. Chaque copie possède ses propres données et s'appelle une image [12].

Les fonctions intrinsèques `num_images()` et `this_image()` permettent de connaître le nombre d'images et le numéro (appelé co-indice) de l'image courante. La syntaxe est simple et inspirée des notations tensorielles [33]. Alors que les indices d'un tableau Fortran sont entre parenthèses, le co-indice est entre crochets. En l'absence des crochets, on accède au co-tableau de l'image courante. Avec les crochets, on accède aux co-tableaux des autres images, de façon transparente. Quand nécessaire, il est possible de synchroniser les images, avec par exemple l'instruction `sync all`.

Voici un exemple simple permettant de calculer une approximation de π par une méthode Monte Carlo (algorithme inefficace mais parfait pour cet exemple !). On déclare un co-tableau `k[*]` d'entiers 64 bits [13] qui est initialisé à zéro dans chaque image puis incrémenté quand un point tiré au hasard dans un carré de côté 2 tombe dans le disque inscrit de rayon 1. L'étoile entre crochets signifie que le nombre de co-dimensions ne sera connu qu'à l'exécution. Une fois que toutes les images sont arrivées à l'instruction `sync all`, l'image 1 se charge de récupérer et de sommer les résultats de toutes les images, pour enfin afficher l'approximation de π.

```
program pi_monte_carlo_coarrays
   use, intrinsic :: iso_fortran_env, only: wp=>real64, int64
   implicit none
   real(wp)       :: xy(2)
   integer(int64) :: i, j, kt=0, k[*]=0
```

12. *Single program, multiple data (SPMD).*

13. Il s'agit ici de scalaires pour simplifier l'exemple, mais un co-tableau peut bien sûr être déclaré à partir d'un tableau classique.

```
    integer(int64), parameter :: n=1000000000

    do i = 1, n / num_images()
    call random_number(xy)
    if (sum(xy**2) < 1) k = k + 1
    end do

    sync all
    if (this_image() == 1) then
    do j = 1, num_images()
    kt = kt + k[j]
    end do
    print *, (4.0_wp * kt) / n
    end if
end program pi_monte_carlo_coarrays
```

Fortran 2008 apporte également les boucles `do concurrent` qui indiquent au compilateur que le code contenu dans une boucle n'utilise pas de données calculées dans des itérations précédentes et qu'elle peut donc être parallélisée si le compilateur juge cela bénéfique. Le compilateur NVIDIA nvfortran est par exemple capable de déporter l'exécution d'une telle boucle sur les GPU des cartes graphiques NVIDIA. Ces boucles finiront par remplacer les boucles parallélisables `forall` qu'avait tenté d'apporter la norme Fortran 95.

Deux nouveautés permettent d'améliorer la structuration du code. Les modules volumineux peuvent être découpés en `submodules`. Et alors qu'auparavant les déclarations devaient être faites avant toute autre instruction, la construction `block...end block` permet de déclarer localement des variables (ou d'importer des modules) en n'importe quel point d'un algorithme. Le code d'une longue procédure peut ainsi être rendu plus lisible.

Parmi les autres apports de la norme Fortran 2008, on notera le module intrinsèque `iso_fortran_env` qui normalise enfin des constantes *(kind)* nommant clairement les types de données (par exemple `real64` et `int64`), le nombre maximum de dimensions d'un tableau (appelé *rank*) qui passe de 7 à 15, la possibilité de lancer un autre programme via la ligne de commande (`execute_command_line`), l'attribut `contiguous` qui indique au compilateur qu'un tableau reçu en argument d'une routine occupera une zone contiguë de la mémoire [14]. Enfin, de nombreuses fonctions intrinsèques sont ajoutées, en particulier pour le calcul scientifique : des fonctions trigonométriques inverses avec des arguments complexes, des fonctions hyperboliques inverses, des fonctions de Bessel, les fonctions erreur et gamma, etc.

14. Ce qui pourra améliorer les performances de la routine.

Fortran 2018

Fortran 2018 apporte de nombreuses améliorations, par exemple sur l'interopérabilité avec le C, mais surtout de nouveaux concepts pour le calcul parallèle. Les images peuvent ainsi être regroupées en équipes *(teams)* travaillant en parallèle sur des tâches différentes. Le concept d'événement *(events)* permet aux images de communiquer simplement entre elles sans avoir recours aux co-tableaux. Quant aux procédures collectives *(collective subroutines),* elles permettent d'effectuer d'une façon simple des opérations de réduction telles que calculer la somme des valeurs dans chaque image d'une variable. Le précédent exemple de calcul de π par Monte Carlo peut ainsi être réécrit d'une façon encore plus concise :

```
program pi_monte_carlo_co_sum
   use, intrinsic :: iso_fortran_env, only: wp=>real64, int64
   implicit none
   real(wp)       :: xy(2)
   integer(int64) :: i, k=0
   integer(int64), parameter :: n=1000000000

   do i = 1, n / num_images()
   call random_number(xy)
   if (sum(xy**2) < 1) k = k + 1
   end do

   call co_sum(k, result_image = 1)
   if (this_image() == 1)  print *, (4.0_wp * k) / n
end program pi_monte_carlo_co_sum
```

Les compilateurs Intel sont les premiers à implémenter intégralement Fortran 2018 et sont depuis fin 2020 librement téléchargeables, comme le reste des *Intel oneAPI Toolkits.* Et avec la version 2023, le compilateur classique ifort passe la main au nouveau compilateur ifx, basé sur l'infrastructure de compilateur LLVM.

Du côté des compilateurs open source, GFortran (GCC) est le plus utilisé. Il n'implémente pas encore intégralement Fortran 2018 et nécessite la bibliothèque OpenCoarrays pour le calcul parallèle, mais l'essentiel y est et le développement progresse.

Fortran et communautés

Dès l'origine

Les langages récents disposent souvent d'une communauté organisée sur internet. Mais Fortran n'a pas attendu les réseaux informatiques pour avoir ses communautés. Ainsi, IBM crée en 1955 le groupe d'utilisateurs SHARE pour ses machines IBM 704. L'équipe Backus y présentera et discutera ses travaux sur le langage et son premier compilateur. Le paragraphe suivant décrit la situation après son déploiement [24] :

> *Néanmoins, la base technique de FORTRAN était suffisamment solide pour que son utilisation fasse boule de neige. Il y eut bientôt des centaines de clients qui firent des centaines de suggestions d'améliorations. Ils trouvaient des bugs et les envoyaient – pas seulement des rapports d'erreur, mais dans de nombreux cas, les corrections étaient envoyées en même temps. De nombreuses suggestions portaient sur des sujets tels que l'amélioration des diagnostics – de petites choses pratiques – et c'était comme si des centaines de personnes travaillaient à l'amélioration de FORTRAN. Les suggestions affluaient et nous les mettions en œuvre aussi vite que possible.*

En 1970 est créé le *Fortran Specialist Group* de la *British Computer Society* [15], qui continue de nos jours ses missions consistant à promouvoir le langage, œuvrer à ses évolutions et favoriser les échanges et rencontres entre utilisateurs.

Usenet

Suite à la création du réseau Usenet, un groupe de discussion `net.lang.f77` est créé en 1983 puis renommé `comp.lang.fortran` en 1986. Il va rester pendant longtemps le principal lieu en ligne où l'on discute Fortran. Il est toujours utilisé, avec typiquement dix nouveaux sujets de discussion par mois. De nos jours, les serveurs Usenet se font rares, y compris dans les universités, mais l'on peut aussi accéder à ces groupes de discussion via Google Groupes.

Quelques sites web de référence

Le Fortran Wiki [16], créé en 2008, est devenu une source importante d'informations sur le langage, en permettant à tout un chacun de l'enrichir.

La page Wikipedia française sur le Fortran est maintenue à jour et offre de nombreuses informations et références sur le langage et son environnement de développement.

Un site [17] créé en 2012 par Ondřej Čertík va longtemps rester une référence dans le monde Fortran. Son contenu a commencé à être transféré vers le site de la communauté Fortran-lang.

15. `https://fortran.bcs.org`.
16. `http://fortranwiki.org`.
17. `https://www.fortran90.org`.

La communauté Fortran-lang

Malgré tous ces outils, il faut reconnaître que la galaxie des utilisateurs de Fortran restait plutôt éparpillée et peu organisée (à part dans les comités de normalisation), comparativement aux langages les plus récents. Pendant longtemps, on trouvait ainsi de nombreux chercheurs ou ingénieurs publiant leurs bibliothèques et programmes isolément sur leurs sites web.

Fin 2019, une poignée de développeurs décident de remédier à cette situation. Ils se regroupent et créent la communauté Fortran-lang en s'appuyant sur la forge logicielle GitHub pour centraliser les développements [26]. Ils ont ainsi créé un site central [18] destiné à offrir toutes les informations et ressources dont peuvent avoir besoin aussi bien le débutant que l'expert. Pour fédérer la communauté, ils ont également mis en place un forum moderne, le Fortran Discourse [19], qui compte désormais plus de 1100 utilisateurs inscrits (de l'étudiant au membre du comité de normalisation Fortran). Les plus impliqués dans l'organisation se retrouvent pour discuter lors de visioconférences mensuelles [20].

La bibliothèque standard stdlib

Ayant réalisé un diagnostic des faiblesses de l'environnement de développement du langage, les fondateurs de la communauté vont rapidement lancer deux projets phares. Contrairement à de nombreux langages (C, C++, Python…), Fortran ne dispose pas d'une bibliothèque standard. Le développement d'une bibliothèque standard *(de facto)* nommée `stdlib` est donc lancé. Il s'agit d'une part d'éviter aux utilisateurs du langage de réinventer la roue chacun de leur côté et d'autre part de servir de laboratoire pour proposer au comité de normalisation de nouvelles fonctionnalités. Sa version 0.2.1 contient déjà des centaines de fonctions utilitaires (chaînes de caractères, fichiers, intégration avec le système d'exploitation, tests unitaires, journalisation…), des algorithmes divers (recherche, tri, fusion...) et des fonctions mathématiques (algèbre linéaire, fonctions spéciales, transformée de Fourier rapide, nombres aléatoires, statistiques, intégration numérique, optimisation...).

Le gestionnaire de paquets fpm

Un autre problème de l'écosystème Fortran était la dispersion des bibliothèques, développées indépendamment et utilisant des outils de constructions variés. Le gestionnaire de paquets `fpm` *(Fortran Package Manager)* y répond [10]. Il s'inspire largement de Cargo, l'utilitaire du langage Rust servant à la fois de gestionnaire de paquets et de système de construction. Créer un projet de type *Hello World*, le compiler et le lancer devient aussi simple que de taper dans un terminal :

18. `https://fortran-lang.org`.
19. `https://fortran-lang.discourse.group`.
20. La communauté a commencé à prendre de l'ampleur à l'époque des premiers confinements COVID.

```
$ fpm new monprojet
$ cd monprojet
$ fpm run
```

Les options du projet sont définies dans son manifeste, un fichier au format TOML présent à sa racine. Le gestionnaire de paquets `fpm` gère les dépendances, qu'il télécharge depuis GitHub. Associé à la commande `git clone`, tester un projet Fortran disponible sur GitHub peut se faire en trente secondes. Les tests du projet peuvent être lancés avec la commande `fpm test`. La communauté promeut en effet les bonnes pratiques actuelles en programmation : tests automatiques, utilisation d'un gestionnaire de versions, etc.

Fpm, dont la version 0.9 est sortie en juin 2023, est désormais un outil très apprécié dans la communauté et progressivement des bibliothèques dites *legacy* sont modernisées en utilisant les dernières fonctionnalités du langage et transformées en paquets `fpm`. Cerise sur le gâteau, `fpm` est essentiellement écrit en Fortran car c'est aussi un langage généraliste. Ce choix audacieux permet en particulier d'attirer plus de contributeurs issus de la communauté et favorise la pérennité du projet.

Nouveau projet phare, le *fpm registry* offrira un dépôt officiel pour les paquets `fpm`. L'utilisateur pourra ainsi facilement découvrir, installer ou publier des paquets Fortran, comme il peut le faire en Python avec PyPI.

Le compilateur interactif LFortran

Un des membres fondateur de la communauté, Ondřej Čertík [21], mène depuis 2019 le développement d'un nouveau compilateur open source LFortran [3], dont l'une des fonctionnalités phares est de pouvoir être utilisé interactivement, par exemple avec Jupyter. Il dispose également de plusieurs *backends* : LLVM, C, C++, Julia, WASM et x86-64. Il n'est pas encore en version bêta mais son développement avance rapidement.

Financements

L'effort de développement de tous ces outils est mené en partie dans le cadre professionnel de certains membres, dans le cadre des études des membres étudiants ou par hobby pour d'autres. Mais il est également possible de financer des efforts supplémentaires. Ainsi, depuis 2021, l'organisation Fortran-lang participe chaque année à la *Google Summer of Code (GSoC)* où plusieurs étudiants payés par Google travaillent sur ses projets [9]. Fin 2022, l'organisme allemand *Sovereign Tech Fund,* dépendant du ministère fédéral de l'économie et du climat, a accordé à l'organisation l'équivalent de trois temps pleins pour travailler six mois sur le gestionnaire de paquets `fpm` et le compilateur LFortran [8]. Il reconnaît ainsi l'importance du langage pour la modélisation du climat.

21. Désormais ingénieur chez GSI Technology, une entreprise fabricant des SRAM.

Conférence internationale FortranCon

Il existe des conférences sur le calcul intensif (*High-Performance Computing* ou *HPC* en anglais) mais une conférence dédiée uniquement au langage Fortran manquait à la communauté. Une première *International Fortran Conference (FortranCon)* a eu lieu à l'université de Zurich du 2 au 4 juillet 2020, en fait virtuellement pour cause de COVID. Elle a permis aux acteurs et utilisateurs du langage de se retrouver et de discuter de leurs travaux et projets. La seconde édition, FortranCon 2021, s'est tenue à nouveau virtuellement à Zurich les 23 et 24 septembre 2021, avec typiquement 70 à 80 participants présents simultanément dans le logiciel de visioconférence Zoom. Les vidéos de ces conférences sont librement accessibles en ligne [14].

Et puis quoi encore ?

Fortran 2023

La norme Fortran 2023 devrait être publiée à l'automne 2023, quasiment 70 ans après la lettre de John Backus à son supérieur. De nombreux compilateurs n'implémentant pas encore intégralement Fortran 2018, la nouvelle norme n'apporte volontairement pas de fonctionnalités majeures mais simplement de nombreuses améliorations dans différentes parties du langage : de nouvelles fonctions pour les chaînes de caractères, en particulier pour faciliter leur passage entre C et Fortran, des fonctions trigonométriques en degrés, des lignes de code source plus longues, de nouveaux descripteurs de format d'affichage, quelques améliorations pour le calcul parallèle, les expressions conditionnelles (même syntaxe qu'en C), etc. L'attribut `simple` permet de définir des fonctions `pure` qui n'accèdent à aucune donnée extérieure si ce n'est par l'intermédiaire de ses arguments. Les boucles `do concurrent` sont désormais capables d'utiliser des variables de réduction. Toutes les nouveautés sont résumées dans le document *The New Features of Fortran 202x* [35].

On peut regretter que le langage n'évolue pas plus vite et qu'il faille attendre longtemps certaines améliorations, là où des langages plus récents ont un développement plus fulgurant. Mais être un langage normalisé a ses avantages et inconvénients, et le comité de normalisation Fortran accorde une attention particulière à la compatibilité ascendante quand de nouvelles fonctionnalités sont ajoutées : ainsi, les compilateurs intégrant les dernières évolutions sont en général toujours capables de compiler les codes dits *legacy,* datant parfois de 40 ans ou plus mais toujours activement utilisés dans de grands organismes ou entreprises.

Notons au passage que les normes ISO sont des documents payants. Mais l'on trouve toujours en accès libre sur le site du comité J3 le document de travail *Fortran 2023 Working Draft* [1]. Tous les travaux du comité y sont d'ailleurs en accès libre.

Fortran 202Y

L'élaboration de la norme suivante, baptisée volontairement de façon vague Fortran 202Y, est en cours. Elle devrait développer considérablement le concept de programmation générique avec en particulier des modèles (*templates*) comme en C++. Il s'agit certainement de la nouveauté la plus attendue par les développeurs Fortran, comme en témoigne le répertoire dédié sur le dépôt GitHub [5] du comité de normalisation J3, où depuis fin 2019 on peut proposer et discuter de nouvelles fonctionnalités pour le langage. Le compilateur LFortran sert d'ailleurs de plateforme d'expérimentation pour l'implémentation de cette future évolution majeure.

Changement important au niveau symbolique, on devrait voir enfin déclaré obsolète l'historique typage implicite par défaut ! Parmi les autres nouveautés dont l'étude est recommandée par le J3, on peut citer la possibilité de définir dans le code lui-même les types utilisés par défaut pour chaque type de données, un préprocesseur Fortran et une gestion des tâches asynchrones.

Conclusion

Au fil de la multiplication des langages, Fortran est devenu un langage de niche. Pour caricaturer (car c'est également un langage généraliste), on pourrait dire qu'il ne fait qu'une seule chose, mais qu'il la fait bien : le calcul numérique [22] ! C'est pourquoi il est toujours largement utilisé dans les calculs météorologiques, la modélisation du climat, la dynamique des fluides, la chimie computationnelle, la modélisation des matériaux [11], l'énergie nucléaire, dans l'aérospatiale, et beaucoup d'autres domaines. Il est même utilisé en économie et en finance [22, 17]. Il s'est adapté à toutes les générations et architectures de matériels informatiques. Le prix Turing 2021 a d'ailleurs été décerné à l'américain Jack Dongarra [23] « *pour ses contributions pionnières aux algorithmes numériques et aux bibliothèques qui ont permis aux logiciels de calcul de haute performance de suivre le rythme des améliorations exponentielles du matériel pendant plus de quatre décennies* » [21].

Dans le domaine du calcul scientifique, Fortran est désormais en concurrence [24] avec de nombreux langages et logiciels : C, C++, Python, Julia, MATLAB, R, etc. Mais le qualifier de vieux langage a-t-il vraiment un sens ? Le manuel de 1956 faisait 54 pages, la norme Fortran 2023 comporte 681 pages, le langage n'ayant cessé d'intégrer de nouveaux paradigmes de programmation. Il est ainsi l'un des rares langages

22. Dans les années 2000, le slogan du défunt compilateur g95 était *it's free crunch time* et celui de GFortran était *free number crunching FORALL!*.

23. Il avait travaillé en particulier sur des bibliothèques Fortran telles que EISPACK (1974), LINPACK, BLAS, LAPACK (1992) et ScaLAPACK (1993), etc. BLAS fait d'ailleurs également partie des dix codes ayant eu le plus d'impact sur la science [34].

24. Tout en se cachant dans les bibliothèques de calcul de certains d'entre eux, par exemple NumPy ou SciPy !

Nom	Adresse internet	Caractéristiques
Fortran Play-Ground	`https://play.fortran-lang.org/`	Par la communauté Fortran-lang ; compilateur GFortran ; librairie stdlib directement accessible
Compiler Explorer	`https://godbolt.org/`	Multi-langage, possibilité de comparer résultats d'exécution et codes assembleur générés par différents compilateurs, possibilité de partage…
tutorialspoint	`https://www.tutorialspoint.com/`	Multi-langage avec matériel didactique ; compilateur GNU ; possibilité de partager publiquement du code
LFortran	`https://dev.lfortran.org/`	Compilateur en version alpha ; affichage graphique ; représentations internes au compilateur et code C++ généré

TABLE 1. Quelques compilateurs Fortran en ligne.

normalisés à intégrer les paradigmes du calcul parallèle [25]. Normalisé depuis bientôt six décennies, ses fondations sont stables tout en lui permettant d'évoluer, ce qui est un gage de pérennité pour l'ingénieur et le chercheur [25]. Côté compilateurs, tous n'intègrent certes pas les dernières normes et certains ont disparu récemment comme le Lahey et l'Absoft, mais d'autres apparaissent. Les ingénieurs d'Intel ont ainsi travaillé cinq ans pour refondre intégralement leur compilateur pour LLVM. Le compilateur Flang open source est également en cours de refonte pour intégrer LLVM, poussé en particulier par NVIDIA. Et un autre nouveau compilateur open source déjà évoqué, LFortran, est en développement.

Enfin, Fortran dispose désormais d'une communauté dynamique et organisée, pleine de projets pour améliorer son environnement de développement et aguerrie aux bonnes pratiques actuelles du développement informatique. Et dix ans après être sorti du top 20 de l'indice TIOBE [2] des langages de programmation les plus « populaires », Fortran y revient périodiquement depuis avril 2021, atteignant même la onzième place en juillet 2023. Le lecteur intéressé trouvera en ligne quelques bons cours en français [4, 15] pour s'initier au langage ou apprendre ses dernières fonctionnalités, du moins jusqu'à Fortran 2008. Pour Fortran 2018, il faudra pour l'instant avoir recours aux livres anglo-saxons cités précédemment [20, 32]. Et il est même possible de s'initier au Fortran moderne en utilisant un des compilateurs en ligne listés Table 1.

Remerciements

Nous remercions Pierre Hugonnet et Jeremie Vandenplas pour leur relecture de l'article et leurs remarques, ainsi que les membres de la communauté Fortran-lang pour la richesse et la qualité des discussions que nous pouvons avoir sur le Fortran

25. Ada 83 était précurseur sur ce point mais les objectifs de fiabilité et de sécurité d'Ada priment sur les performances. Fidèle lui aussi à son impulsion originelle, Fortran aborde le parallélisme en se focalisant sur les performances en calcul numérique.

Discourse, pour la vision de ses fondateurs et pour les efforts accomplis pour améliorer l'environnement de développement du langage.

Références

[1] Brouillon de la norme Fortran 2023 : https://j3-fortran.org/doc/year/23/23-007r1.pdf.

[2] Classement TIOBE des langages les plus populaires : https://www.tiobe.com/tiobe-index/fortran/.

[3] Compilateur LFortran : https://lfortran.org/.

[4] Cours de Fortran de l'IDRIS (Institut du développement et des ressources en informatique scientifique du CNRS) : http://www.idris.fr/formations/fortran/.

[5] Dépôt du comité J3 : https://github.com/j3-fortran.

[6] Earth Simulator : https://www.top500.org/resources/top-systems/the-earth-simulator-earth-simulator-center/.

[7] *Fortran - Programmer's reference manual :* http://archive.computerhistory.org/resources/text/Fortran/102649787.05.01.acc.pdf.

[8] Fortran et le Sovereign Tech Fund : https://sovereigntechfund.de/fortran_en.

[9] Fortran-lang et la Google Summer of Code : https://summerofcode.withgoogle.com/programs/2023/organizations/fortran-lang.

[10] Fortran Package Manager fpm : https://github.com/fortran-lang/fpm.

[11] Logiciel FORGE® pour la simulation des procédés de mise en forme à chaud et à froid : https://www.transvalor.com/fr/forge.

[12] Programmation orientée prototype : https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_prototype.

[13] PVM (Parallel Virtual Machine) : https://www.csm.ornl.gov/pvm/.

[14] Vidéos des conférences FortranCon : https://www.youtube.com/@fortrancon6189.

[15] Introduction au fortran 90/95/2003/2008 : polycopié de Master de Jacques Lefrère, Université Pierre et Marie Curie Sorbonne Université : http://wwwens.aero.jussieu.fr/lefrere/master/mni/f90+c/polyf90.pdf, November 2019.

[16] Frances ("Fran") Elizabeth Allen. A.M. Turing Award 2006 : https://amturing.acm.org/award_winners/allen_1012327.cfm.

[17] Antoine Arnoud, Fatih Guvenen, and Tatjana Kleineberg. Benchmarking Global Optimizers. Technical Report w26340, National Bureau of Economic Research, Cambridge, MA, October 2019.

[18] J. W. Backus, R. J. Beeber, S. Best, R. Goldberg, L. M. Haibt, H. L. Herrick, R. A. Nelson, D. Sayre, P. B. Sheridan, H. Stern, I. Ziller, R. A. Hughes, and R. Nutt. The FORTRAN Automatic Coding System. In *Papers Presented at the February 26-28, 1957, Western Joint Computer Conference : Techniques for Reliability*, IRE-AIEE-ACM '57 (Western), page 188–198, New York, NY, USA, 1957. Association for Computing Machinery.

[19] John Backus. A.M. Turing Award 1977 : https://amturing.acm.org/award_winners/backus_0703524.cfm.

[20] Milan Curcic. *Modern Fortran : Building efficient parallel applications*. Manning Publications, 1re edition, November 2020.

[21] Jack Dongarra. A.M. Turing Award 2021 : https://amturing.acm.org/award_winners/dongarra_3406337.cfm.

[22] Hans Fehr and Fabian Kindermann. *Introduction to Computational Economics Using Fortran*. Oxford University Press, May 2018.

[23] Virginia Grant. We thank Miss Mary Tsingou, December 2020.

[24] W. P. Heising. The Emergence of FORTRAN IV from FORTRAN II. In *Annals of the History of Computing*, volume 1, pages 28–32, January 1984.

[25] K. Hinsen. Dealing With Software Collapse. *Computing in Science Engineering*, 21(3) :104–108, 2019.

[26] Laurence J. Kedward, Balint Aradi, Ondrej Certik, Milan Curcic, Sebastian Ehlert, Philipp Engel, Rohit Goswami, Michael Hirsch, Asdrubal Lozada-Blanco, Vincent Magnin, Arjen Markus, Emanuele Pagone, Ivan Pribec, Brad Richardson, Harris Snyder, John Urban, and Jeremie Vandenplas. The State of Fortran. *Computing in Science & Engineering*, 24(2) :63–72, March 2022.

[27] Ken Kennedy, Charles Koelbel, and Hans Zima. The rise and fall of High Performance Fortran : an historical object lesson. In *Proceedings of the third ACM SIGPLAN conference on History of programming languages*, San Diego California, June 2007. ACM.

[28] Mark Jones Lorenzo. *Abstracting Away the Machine : The History of the FORTRAN Programming Language (FORmula TRANslation)*. SE Books, August 2019.

[29] Vincent Magnin, James Tappin, Jens Hunger, and Jerry DeLisle. gtk-fortran : a GTK+ binding to build Graphical User Interfaces in Fortran. *The Journal of Open Source Software*, February 2019.

[30] Frédéric Magoulès and François-Xavier Roux. *Calcul scientifique parallèle*. October 2017.

[31] Arjen Markus. *Modern Fortran in Practice*. Cambridge University Press, Cambridge, 2012.

[32] Michael Metcalf, John Ker Reid, and Malcolm Cohen. *Modern Fortran explained : incorporating Fortran 2018*. Numerical mathematics and scientific computation. Oxford University Press, 2018.

[33] Robert W. Numrich. Parallel numerical algorithms based on tensor notation and Co-Array Fortran syntax. *Parallel Computing*, 31(6) :588–607, June 2005.

[34] Jeffrey M. Perkel. Ten computer codes that transformed science. *Nature*, 589(7842) :344–348, January 2021.

[35] John Reid. The New Features of Fortran 202x. https://wg5-fortran.org/N2151-N2200/N2194.pdf. Technical report, 2022.

[36] Damian Rouson, Jim Xia, and Xiaofeng Xu. *Scientific software design : the object-oriented way*. Cambridge Univ. Press, Cambridge, 1. pbk. ed edition, 2014.