EXERCICES COMMENTÉS DE PROGRAMMATION EN LANGAGE FORTRAN

A L'USAGE DES DÉBUTANTS

PAR

Marc THORIN

MASSON et C^ie^, ÉDITEURS
120, B^d^ Saint-Germain, Paris (6^e^)
1972

AVANT-PROPOS

Ce livre n'est pas un ouvrage de référence, mais un petit guide pour ceux qui commencent à programmer.

Ce n'est pas un cours — malgré quelques rappels occasionnels — mais un livre d'exercices.

Le FORTRAN *peut-il être appris dans les livres ?*

Au point d'être maîtrisé, certainement pas. La présence d'un programmeur bon pédagogue et surtout la pratique de l'ordinateur sont indispensables. Alors, pourquoi cet ouvrage ?

Parce que d'une part, l'on dispose rarement de l'un et de l'autre autant qu'il serait souhaitable, que d'autre part, le Fortran est comme une langue étrangère : s'il est vain d'espérer l'apprendre à fond dans un livre, celui-ci peut donner au débutant des bases solides et préparer la rencontre avec l'ordinateur — assez brutale et décevante pour qui est trop novice.

Ces exercices n'ont pas d'autre but. Il est à espérer que le lecteur aura vite dépassé leur niveau.

Leur intérêt scientifique est nul : parce que leur rôle n'est que pédagogique ; parce qu'ils ne doivent faire appel qu'à des notions élémentaires de mathématiques, toute théorie étant écartée ; parce qu'ils doivent être courts et si possible amusants, pour ne pas rebuter, et pourtant complets pour bien montrer leur organisation générale, au risque de n'être pas compris intégralement par le débutant qui ne connaît pas encore tous les ordres — qu'importe, il n'a qu'à laisser provisoirement des « blancs ».

Les corrigés sont suivis de l'indication d'erreurs couramment faites : il faut montrer à l'élève pourquoi il est tombé et le relever tout de suite ; la présentation de certains énoncés est destinée à révéler des fautes immédiatement corrigées : il se méfiera plus la prochaine fois. La programmation n'est pas l'affaire des gens distraits. Mais il faut graduer les difficultés sinon le résultat serait à l'opposé de ce qui est voulu : l'élève n'oserait plus faire un pas.

Les exemples, classés dans chaque chapitre par difficulté croissante, sont sauf mention contraire indépendants ; certaines erreurs sont donc signalées plusieurs fois (d'ailleurs enseigner c'est répéter), d'autres n'ont pas été indiquées dès le début pour alléger le corrigé des premiers programmes. Beaucoup d'ordres Fortran ne sont pas mentionnés dans les exercices ; mais ce sont des ordres mineurs quoique parfois très utiles, et ceci n'est pas un glossaire mais est destiné à faire comprendre l'utilité du Fortran. Pour qui connaît solidement ses bases, se mettre au courant de l'ordre DATA *ou des calculs en complexes par exemple ne demande que cinq minutes ; alors que celui qui a une connaissance générale de tous les ordres mais une compréhension approximative du Fortran ne peut écrire un bon programme. Dans le même esprit, on n'a pas parlé des facilités qu'offrent certains compilateurs, au point de présenter comme erreur ce qui passerait sur beaucoup de machines : lorsque l'on apprend une langue, c'est avec un purisme qui n'est pas courant en réalité. Il est toujours assez tôt pour connaître les tolérances usuelles.*

TABLE DES MATIÈRES

INTRODUCTION

DÉTAILS TECHNIQUES

Les programmes sont écrits pour passer en principe sur tous les ordinateurs ayant un compilateur Fortran. Cependant, il faut bien fixer certains détails.

L'ordinateur considéré a 8 chiffres décimaux significatifs tant pour les entiers que pour la mantisse des réels.

Il dispose d'un lecteur (référencé 1) et d'une imprimante (référencée 2). Tous les formats de sortie commencent donc par 1 Hb, caractère de saut d'une ligne, toujours isolé des codes de conversion et même des autres formats H pour habituer l'élève à faire attention à ce caractère.

Le nombre de caractères par ligne de programme est quelconque, pour l'imprimante il n'est pas défini, mais on met en garde lorsqu'il dépasse 100.

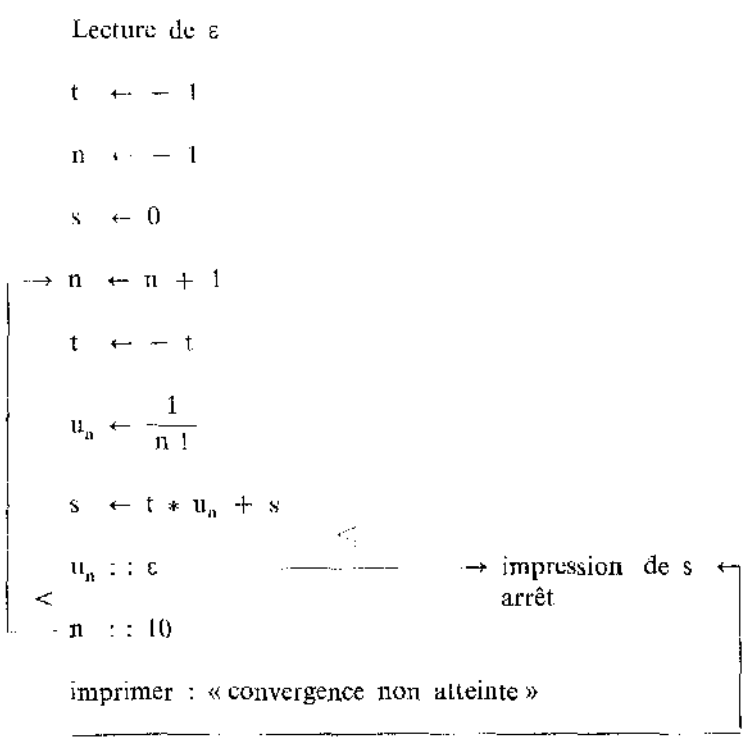
La machine ne fait aucune conversion implicite, les opérations ne sont admises qu'entre nombres de même nature. Quoique, pour la plupart, les compilateurs l'admettent, cela a été voulu par souci de généralité et surtout de pédagogie.

L'indice des variables doit être tout calculé.

Les expressions et le IF logiques sont admis en raison de leur importance et pour avoir l'occasion d'être appris, quoique certains compilateurs les refusent.

ORGANIGRAMMES

Les organigrammes ne sont pas présentés sous les formes courantes, qui ne sont pas vraiment normalisées, mais d'une façon condensée et plus rapide à écrire. Les conventions vont être expliquées sur le calcul de $\Sigma \frac{(-1)^n}{n!}$, n partant de 0, jusqu'à ce que $\left| \frac{(-1)^n}{n!} \right|$ soit inférieur à un ε donné, ou à défaut que n soit égal à 10.



Le signe ← indique que la valeur calculée à droite est affectée à la variable indiquée.

u_n : : ε signifie que u_n est comparé à ε et l'ordre désigné par la flèche est effectué si $u_n \leqslant \varepsilon$.

Sinon, l'ordre de la ligne suivante est exécuté. On s'efforce par souci de lisibilité — mais ce n'est pas toujours possible — d'écrire les ordres en colonne, quitte à mettre n ← − 1 et tout de suite après n ← n + 1.

TROIS EXEMPLES

Le premier explique le passage de l'algorithme à l'organigramme puis au programme et veut mettre en garde le débutant contre le penchant naturel à attendre de l'ordinateur un comportement tant soit peu intelligent. La bonne compréhension des étapes qui conduisent au programme est indispensable pour bien situer le travail du programmeur et le rôle de la machine.

La traduction d'un organigramme donné constitue le second exemple.

Le troisième est l'écriture et la programmation complète d'un exemple très court mais où quelques pièges sont tendus.

Premier exemple : racine carrée d'un nombre donné A.

Soit un nombre donné A dont on se propose de calculer la racine carrée.

a) **La méthode, expliquée pour un être humain : algorithme.** — On prend un nombre quelconque u_0 et l'on calcule successivement :

$$u_1 = \frac{1}{2}\left(u_0 + \frac{A}{u_0}\right)$$

$$u_2 = \frac{1}{2}\left(u_1 + \frac{A}{u_1}\right)$$

— — — — —

$$u_{n-1} = \frac{1}{2}\left(u_{n-2} + \frac{A}{u_{n-2}}\right)$$

$$u_n = \frac{1}{2}\left(u_{n-1} + \frac{A}{u_{n-1}}\right)$$

— — — — —

On démontre que les u_n forment une suite de nombres qui se rapprochent de plus en plus de $\sqrt{A}$.

A ce stade, avoir compris (indépendamment de tout développement mathématique) la méthode, suppose :

— de l'imagination, puisque toutes les formules n'ont pas été écrites,

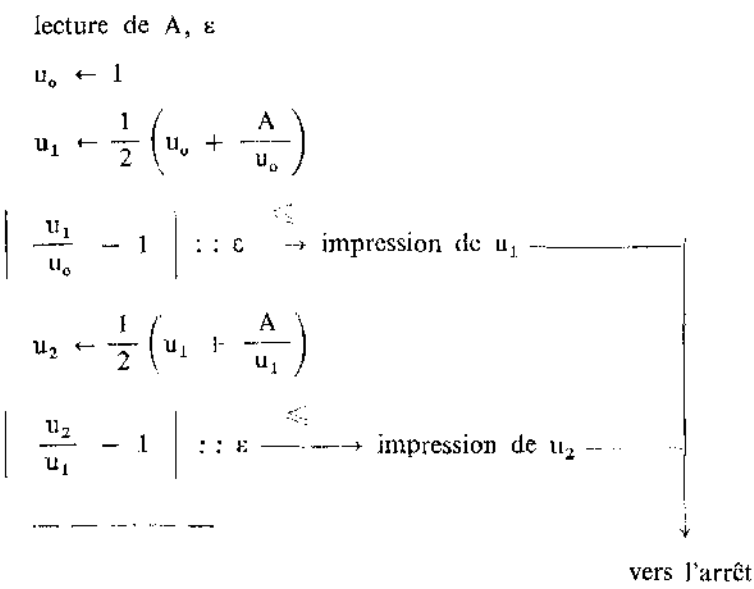
— de l'abstraction, puisque u_0 et A ne sont pas précisés,

— de la mémoire et des connaissances (bien qu'élémentaires) de mathématiques pour savoir ce que signifient les formules,

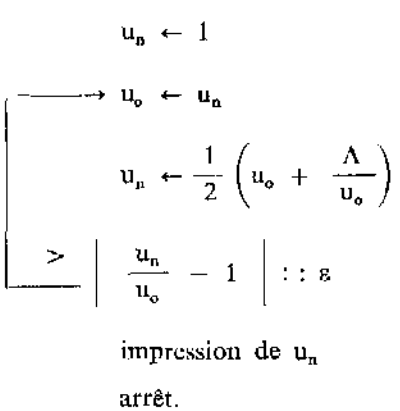
— du flair, puisque le test d'arrêt des calculs n'est pas défini.

Toutes ces qualités sont proprement humaines. Pour un automate, et en particulier un ordinateur, il n'est pas question de comprendre mais seulement d'exécuter mécaniquement un travail parfaitement défini.

b) **La méthode exposée pour un automate : organigramme.**



Il a fallu préciser A, u_0, et le test. Pour des raisons pratiques, on ne peut écrire une suite infinie d'ordres. Il faut faire une boucle, et imaginer un décalage :



Chaque opération est bien définie. Il suffit à l'automate de posséder un système de perception de ces ordres et de pouvoir

les exécuter, c'est-à-dire d'avoir une mémoire contenant des constantes et des organes de calcul excités par la perception des ordres.

c) **La méthode, écrite pour un ordinateur : programme.** — L'ordinateur électronique actuel est un cas particulier d'un tel automate. Ce qui suit n'est plus que technologie :

les ordres suivants, codés en Fortran, écrits sur un support convenable (carte), lus par un certain périphérique (lecteur), provoquent une succession d'états amenant la valeur voulue dans la mémoire désignée par UN, dont le contenu est alors transféré sur un support externe (papier de l'imprimante).

```
C       RACINE CARREE D'UN NΦMBRE DΦNNE A
        READ(1,100) A,EPS
100     FΦRMAT(E15.7)
        UN=1.
10      UZ=UN
        UN=0.5*(UZ+A/UZ)
        IF(ABS(UN/UZ - 1.)—EPS) 20,20,10
20      WRITE(2,200) UN
200     FΦRMAT(1H ,E20.7)
        STΦP
        END
```

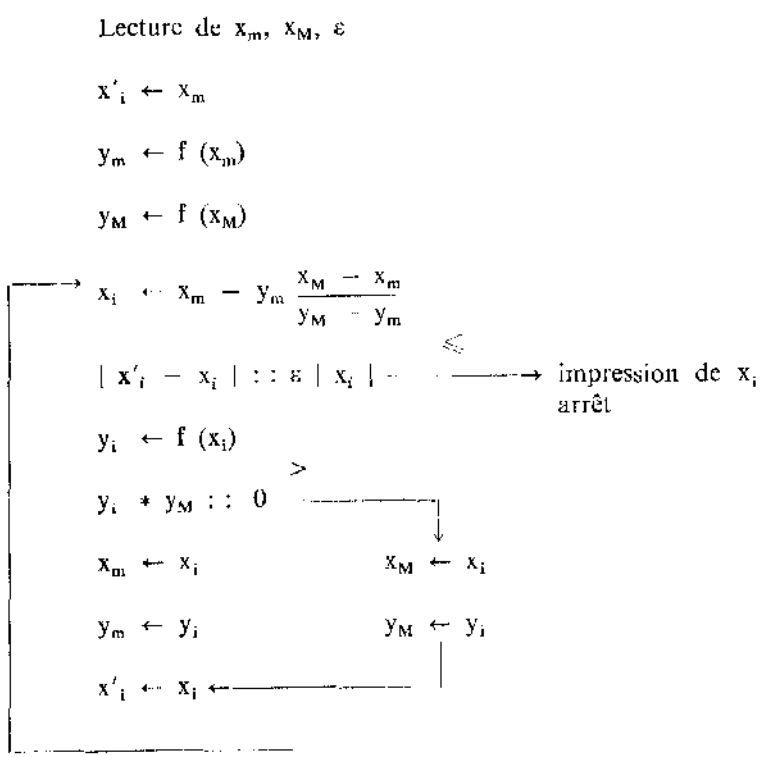
Dans ces conditions, attendre la moindre intelligence, le moindre sens critique, la moindre initiative d'un ordinateur est une absurdité.

On ne peut passer sous silence l'existence du compilateur, programme qui traduit le programme écrit en Fortran en une suite des opérations élémentaires à effectuer. Les ordres tels que DIMENSIΦN, END, etc. lui sont destinés.

Deuxième exemple : calcul des racines d'une fonction par la méthode de Newton

Organigramme. — Voici l'organigramme du calcul de la racine d'une fonction f(x) comprise entre les abscisses x_m et x_M.

Le traduire en Fortran en prenant $f(x) = \cos x + 0.5\, e^x$. Il est inutile, pour notre propos qui est d'apprendre le Fortran de comprendre la méthode d'un point de vue mathématique. Il faut viser ici simplement une transcription sans faute, sans chercher d'astuce ni prendre d'initiative.



Programme :

```
C       CALCUL DES RACINES D'UNE FONCTION
        F(X)=COS(X)+0.5*EXP(X)
        READ(1,200) XMIN,XMAX,EPS
200     FORMAT(E15.7)
        XPI=XMIN
        YMIN=F(XMIN)
        YMAX=F(XMAX)
1       XI=XMIN-YMIN*(XMAX-XMIN)/(YMAX-YMIN)
        IF(ABS(XPI-XI)-EPS*ABS(XI)) 10,10,20
10      WRITE(2,100) XI
100     FORMAT(1H ,E15.7)
        STOP
20      YI=F(XI)
        IF(YI*YMAX) 30,30,40
40      XMAX=XI
```

```
        YMAX=YI
        GϕTϕ 123
30      XMIN=XI
        YMIN=YI
123     XPI=XI
        GϕTϕ 1
        END
```

Exemples d'erreur et de maladresse :

- Oublier le signe * pour multiplier,
- Omettre des parenthèses nécessaires (erreur) ou en mettre trop sans tenir compte de la hiérarchie des opérateurs (maladresse, coûtant cher en temps de calcul),
- S'efforcer de numéroter les adresses dans l'ordre,
- Ne pas définir en tête la fonction,
- Croire que la place des FϕRMAT a une importance,
- Renvoyer (par un GϕTϕ ou autrement) à l'adresse placée devant l'un des FϕRMAT
- Mettre des indices ou des « primes » dans les noms de variables,
- Se tromper dans l'ordre des adresses suivant un IF,
- Ne pas mettre un STϕP après l'impression de XI; en mettre un après le GϕTϕ 1,
- Oublier le END.

Troisième exemple : calcul d'une trajectoire point par point

Ecrire l'organigramme puis le programme donnant les ordonnées y, pour les abscisses x croissant de 100 en 100 à partir de 100 jusqu'à ce que y devienne négatif.

Appliquer la formule :

$$y = -\frac{1}{2}\, g\, \frac{x^2}{V_0^{\,2}\cos^2\lambda} + x \operatorname{tg}\lambda$$

où

$g = 9.81$ m/s^2

V_0, vitesse initiale, est lue

λ, angle de tir est lu mais donné en millièmes

(6400 millièmes = 2 π radians).

On donne $\pi = 3.14159265358979$

On veut le maximum de précision, indépendamment de toute considération physique.

Les buts de cet exemple sont :

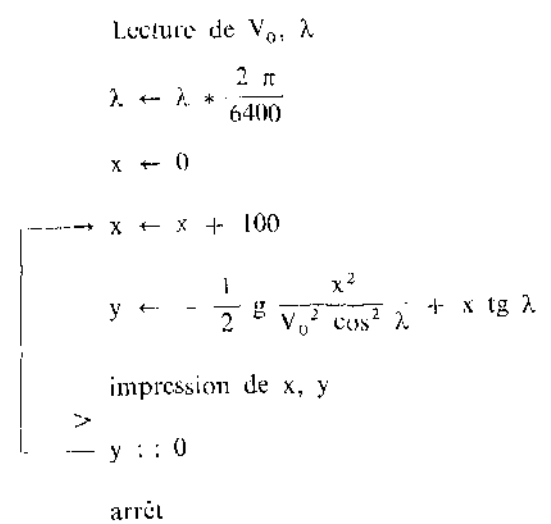
— premièrement, de faire non plus seulement lire mais écrire

un organigramme, ce qui suppose une parfaite compréhension du problème et de la méthode;

- deuxièmement, de réunir plusieurs conventions du Fortran qui ne présentent apparemment pas de difficulté, mais où les débutants tombent toujours, et qui rendent tout travail sérieux impossible si elles ne sont pas très bien assimilées.

Cet exercice et la liste de corrections sont très importants.

Organigramme :



Programme :

```
C       CALCUL D'UNE TRAJECTOIRE
        READ(1,100) VZ,ZLAMB
100     FORMAT(2E15.7)
        ZLAMB=ZLAMB/3200.*3.1415927
        X=0.
1       X=X+100.
        Y=-9.81/2.*(X/VZ/COS(ZLAMB))**2+X*TAN(ZLAMB)
        WRITE(2,200) X,Y
200     FORMAT(1H ,2E20.7)
        IF(Y) 2,1,1
2       STOP
        END
```

Exemples d'erreur et de maladresse :

• Chercher à mettre un indice à V, ou l'appeler VO : la faute de frappe ϕ pour O est difficile à déceler,

• Appeler λ : LAMBDA, variable entière (règle I,J,K,L,M,N),

• Donner à λ un autre nom après sa conversion,

• Ne pas effectuer $\frac{2}{6400}$,

• Prendre π avec toutes les décimales données alors qu'il n'y a que 8 chiffres significatifs en machine,

• Prendre $\pi = 3.14$, avec le même nombre de chiffres significatifs que g : cette règle, utile pour le calcul manuel, où l'on ne veut pas se donner de la peine pour un gain assez illusoire, n'a pas de raison d'être en machine où le nombre de chiffres significatifs est bien défini et où les opérations prennent donc un temps fixe : la précision ne coûte rien,

• Croire que la formule ZLAMB = ZLAMB / 3200. * 3.1415927 est la transcription de $\frac{\lambda}{3200\ 11}$ (règle de hiérarchie),

• Ne pas mettre de point après le O dans l'écriture de X, puisqu'un réel qui tombe juste n'a rien à voir avec un entier,

• Mettre des parenthèses X / (VZ * COS (ZLAMB)) (maladresse) ou écrire X / VZ * COS (ZLAMB) (faux),

• Mettre un point après l'exposant 2 (appel des deux fonctions EXP et ATOG au lieu d'une multiplication),

• Multiplier explicitement $\frac{x}{V_0 \cos \lambda}$ par lui-même au lieu d'élever au carré,

• Poser A = − 9.81/2. et utiliser cette valeur dans le calcul (un appel en mémoire est plus long qu'une division de deux constantes). D'ailleurs, ce coefficient devrait être calculé à la main,

• Se tromper dans l'ordre des adresses après le IF arithmétique,

• Mettre des codes de conversion compliqués en entrée. Ne pas espacer les résultats en sortie,

• Confondre ou oublier STOP et END,

• Ecrire 1/2 (donne 0 en entier) ou 1./2. (très maladroit).

• Appeler λ : ZLAMBDA (plus de 6 caractères).

CONSEILS

• Faites très attention à l'écriture exacte des ordres : l'ordinateur est un automate, et mettre un point de trop, omettre un astérisque par exemple, revient à se tromper de levier sur une

machine : l'ordinateur est... pointilleux et ne saura pas vous corriger, mais tout au plus signaler qu'il ne peut exécuter l'ordre.

• Programmer demande une forme d'esprit qui n'est pas innée : ne vous acharnez pas sur des détails d'un exemple dès le début. Passez à un autre : c'est très simple et vous comprendrez en y revenant une autre fois. Mais il faut y revenir, relaire les programmes quelques jours après avoir vu le corrigé. Il y a peu d'exercices, ils doivent être traités en profondeur.

• Prenez des noms mnémotechniques pour vos variables, même s'ils sont plus longs. Quand vous ne savez plus où vous en êtes dans le décompte de vos adresses, passez carrément à la centaine supérieure. Tout cela ne prend absolument pas de place supplémentaire en mémoire. Vous pouvez, par exemple, réserver les dizaines justes pour les DO et les centaines pour les FORMAT.

• Faites preuve d'astuce : réduisez la longueur de votre programme, le nombre des variables, le nombre des adresses, le temps de calcul (tout cela va en général simultanément).

Les exercices de ce livre sont courts, l'avantage se sentirait peu en général, mais sur un programme réel on peut faire un gain considérable par rapport à la première version, ou même... éviter d'avoir à changer de machine.

De plus, cela force à réfléchir et fait découvrir bien des fautes. Par exemple, pour calculer $\frac{a}{bc}$, la solution paresseuse qui dispense d'avoir compris, est de mettre A/(B*C) au lieu de A/B/C.

Toutefois, les corrigés s'adressent à des débutants; on y a évité les astuces de logique trop difficiles et celles qui tiennent au Fortran lui-même (voir cependant à titre d'exemple le sous-programme de factorielle).

• Estimez, même si l'on ne vous le demande pas, le volume des résultats, le temps de calcul et l'encombrement en mémoire de vos programmes, lorsque vous aurez un peu l'habitude d'une machine donnée. Cela vous évitera des mésaventures.

• La valeur de π n'est pas en mémoire quoiqu'elle soit très fréquemment utilisée.