H. LEHNING D. JAKUBOWICZ

mathématiques par l'informatique individuelle

1

le basic
arithmétique • cryptographie • équations

2e édition

MASSON

mathématiques par l'informatique individuelle

1

le basic

arithmétique • cryptographie • équations

CHEZ LE MÊME ÉDITEUR

Des mêmes auteurs

MATHÉMATIQUES PAR L'INFORMATIQUE INDIVIDUELLE, par H. LEHNING et D. JAKUBOWICZ.
Tome 2. — Approximation, sommation, 1982, 128 pages.
Tome 3. — Optimisation, simulation, statistique. *En préparation.*
Tome 4. — Graphisme. 1984, 112 pages.

LE BASIC ILLUSTRÉ, par D. ALCOCK. 1983, 2e tirage, 144 pages.

MATHÉMATIQUES APPLIQUÉES ET CALCULATRICES PROGRAMMABLES. Notation algébrique (TI 57, 58, 59), par L. SOLOMON et M. HOCQUEMILLER. 1982, 264 pages.

CALCUL SYMBOLIQUE ET INFORMATIQUE. Du calcul numérique au calcul littéral. Programmes en Basic, par A. DESHAYES. *Collection Méthodes + Programmes.* 1985, 184 pages.

ALGORITHMIQUE ET REPRÉSENTATION DES DONNÉES. *Collection Manuels Informatiques Masson.*
Tome 1. — Files, automates d'états finis, par M. LUCAS, J.-P. PEYRIN et P.-Cl. SCHOLL, 1985, 2e tirage, 200 pages.
Tome 2. — Évaluation, arbres, graphes, analyse de textes, par M. LUCAS. 1984, 160 pages.
Tome 3. — Recursivité et arbres, par P.-Cl. SCHOLL. 1984, 224 pages.

GRAPHISME SCIENTIFIQUE SUR MICRO-ORDINATEUR, de la 2e à la 3e dimension. 50 applications résolues en Basic, par R. DONY. *Collection Méthodes + Programmes.* 1985, 2e édition, 256 pages.

ABC DE CRYPTOGRAPHIE, avec programmes en Basic, par R. ROUBATY. *Collection Méthodes + Programmes.* 1985, 208 pages.

CALCUL ASTRONOMIQUE POUR AMATEURS adapté à l'emploi d'un calculateur ou d'un micro-ordinateur, par S. BOUIGES. 1985, 4e édition, 2e tirage, 168 pages.

mathématiques par l'informatique individuelle

1

le basic

arithmétique • cryptographie • équations

HERVÉ LEHNING
Professeur de Mathématiques Spéciales
Paris

DANIEL JAKUBOWICZ
Professeur de Mathématiques Spéciales
Versailles

2e édition
révisée et complétée

MASSON
Paris New York Barcelone Milan
Mexico Rio de Janeiro São Paulo
1985

ISBN : 2-225-80697-7

MASSON S.A.	120, bd Saint-Germain, 75280 Paris Cedex 06
MASSON PUBLISHING U.S.A. Inc.	1 Ames Court, Plainview, N.Y. 11803
MASSON S.A.	Balmes 151, Barcelona 8
MASSON ITALIA EDITORI S.p.A.	Via Giovanni Pascoli 55, 20133 Milano
MASSON EDITORES	Dakota 383, Colonia Napoles, Mexico 18 D.F.
EDITORA MASSON DO BRASIL Ltda	Rua Borges Lagoa 1044, CEP/04038 Sao Paulo S.P.

Avant-propos

Les mathématiques par l'informatique individuelle. Certains — ne voyant que la partie "démonstrative" des mathématiques — verront dans ce titre un paradoxe : celle-ci n'a (pour l'instant) rien à faire de l'informatique individuelle. Mais les mathématiques ne sont pas que démonstratives :

— elles sont intuitives et expérimentales car avant de démontrer un théorème, il faut avoir l'intuition de son énoncé ,

— elles sont appliquées car un résultat théorique ne débouchant pas sur une application concrète n'a que peu d'intérêt.

Cette série d'ouvrages traite donc de ces deux aspects des mathématiques.

Ce premier tome introduit la notion d'algorithme informatique et le langage Basic puis discute la performance de ceux-ci du point de vue de la taille mémoire utilisée et du temps de calcul à travers l'exemple de la décomposition en facteurs premiers. Je peux ensuite aborder les calculs approchés qui introduisent une nouvelle contrainte sur les algorithmes utilisés : l'erreur sur le résultat obtenu.

Les deux dernières parties traitent de la résolution des équations et des systèmes d'équations numériques, les problèmes d'erreurs y sont systématiquement discutés et les programmes de résolution sont appliqués à des problèmes d'origine concrète.

Pour des raisons techniques, les listes de programme ont été éditées sur l'imprimante d'un ordinateur HP 85 . Toutefois, ceux-ci ont été rédigés dans la version la plus simple de Basic, et testés sur des ordinateurs de poche.

Je remercie Messieurs Didier MONTERNOT et Alexandre OCAÑA pour l'aide technique qu'ils m'ont apportée et Hélène qui n'a pas ménagé sa peine.

Enfin, je remercie Chantal pour avoir vérifié certains calculs et Corinne à qui je dédie cet ouvrage.

Pour cette seconde édition, je tiens à remercier les lecteurs qui m'ont permis de l'améliorer et, en particulier, les groupes des IREM de Lyon et de Montpellier.

Hervé LEHNING
Bourg-la-Reine, mai 1985

Table des matières

Première partie

ALGORITHMES INFORMATIQUES ET LANGAGE BASIC

Deuxième partie

ARITHMETIQUE ET APPLICATION À LA CRYPTOGRAPHIE

Troisième partie

LE CALCUL APPROCHÉ

Quatrième partie

LA RÉSOLUTION DES ÉQUATIONS

Cinquième partie

LA RÉSOLUTION DES SYSTÈMES D'ÉQUATIONS

PREMIERE PARTIE

ALGORITHMES INFORMATIQUES

ET

LANGAGE BASIC

Je n'infligerai pas au lecteur une définition du mot "algorithme" ni - et encore moins - une du mot "langage".

Le meilleur *synonyme d'algorithme* est sans doute "recette" : la liste des opérations à effectuer pour obtenir le résultat désiré (comme en cuisine).

Je vais préciser cette notion ainsi que sa mise en oeuvre sur l'étude de trois exemples :

- le calcul du nombre de jours entre deux dates,
- les calculs arithmétiques sur les nombres rationnels,
- les calculs sur les polynômes.

Exemples choisis pour répondre aux trois critères suivants :

- permettre une rencontre progressive des instructions fondamentales du BASIC,
- ne pas poser de problèmes quant à la validité des résultats : il s'agit essentiellement de calculs exacts,
- et, bien entendu, avoir un intérêt pratique.

CHAPITRE 1

LE CALCUL DU NOMBRE DE JOURS ENTRE DEUX DATES

Le résultat désiré est de calculer le nombre de jours entre deux dates, par exemple entre le 23 Février 1961 et le 10 Avril 1985.

Cela peut paraître n'avoir qu'un intérêt de simple curiosité mais il en existe de *nombreuses applications* notamment :

- *calcul de biorythmes*
  voir le paragraphe 3 de ce chapitre,

- *calcul des positions planétaires*
  voir le chapitre 8 de la quatrième partie.

Je me limiterai au calendrier grégorien, en usage en occident depuis 1582.

Avant de passer à l'étude de la programmation elle-même, j'en rappelle les caractéristiques - ce qui est nécessaire pour une analyse correcte du problème.

1. ANALYSE ET PROGRAMMATION

A. LE CALENDRIER GREGORIEN

Comme chacun sait, *l'année grégorienne* est divisée en 12 mois, chacun ayant le nombre de jours donné dans le tableau :

Janvier	31	Février	28	Mars	31	Avril	30
Mai	31	Juin	30	Juillet	31	Août	31
Septembre	30	Octobre	31	Novembre	30	Décembre	31

ce qui donne 365 jours pour une année.

L'année tropique (durée d'une révolution de la terre autour du soleil) est en fait de 365,24220 jours à 10^{-5} près.

Pour éviter un décalage des saisons par rapport au soleil (ce qui est gênant en agriculture), Jules César a décidé d'ajouter un jour tous les quatre ans : le 29 Février des années - dites bissextiles - dont le millésime est divisible par 4.

La différence de l'année julienne moyenne avec l'année tropique est donc de 0,00780 jour soit environ 3 jours tous les 400 ans.

Donc, toujours pour éviter un décalage avec les saisons, le pape Grégoire a décidé que les années - dites séculaires - dont le millésime est divisible par 100 ne sont pas bissextiles sauf si leur millésime est divisible par 400.

L'année grégorienne moyenne est donc de 365,24250 jours ce qui donne une différence de 3 jours en 10.000 ans sur l'année tropique : la réforme du calendrier n'est donc pas complètement achevée.

Il est maintenant possible d'analyser le problème :

B. ANALYSE GENERALE DU PROBLEME

- Mémorisation de la date.

Elle nécessite trois variables numériques J,M et A (une *variable numérique* est une case mémoire destinée à contenir des nombres) pour être mémorisée.

- Pour éviter de tenir compte des années séculaires, je peux me limiter aux années de 1901 à 2099 (2000 est divisible par 400 !). Je remarque alors que le problème est plus simple si je calcule le nombre de jours écoulés entre une date et la première de mon domaine de validité : le 0 Janvier 1901 (jour fictif) plutôt que le 1er Janvier 1901 par commodité.

- Le coeur du programme sera donc constitué par un *module de calcul* du nombre de jours écoulés entre une date et le 0 Janvier 1901. Il comportera en entrée une date (variables J,M et A) et en sortie un nombre (variable N).

A ce stade, la logique de déroulement du programme est dégagée.

C. LOGIQUE DE DEROULEMENT DU PROGRAMME

Le schéma ci-après s'appelle un *organigramme*, il décrit la suite des opérations que doit exécuter l'ordinateur de la façon suivante :

L'exécution commence par *l'ovale* DEBUT et finit par l'ovale FIN ; entre les deux elle suit les flèches.

Ici, vous rencontrez dans l'ordre :

• un *parallélogramme* (désigne une entrée de données ou une sortie de résultats) contenant la phrase :

"Introduire la première date j,m,a → J,M,A"

ce qui signifie que vous devez taper les nombres j,m,a au clavier et que l'ordinateur doit mettre ces valeurs dans les cases mémoires J,M,A. Ce qui se dit de façon plus savante :

les valeurs j,m,a *sont affectées aux variables* J,M,A

ou :

les variables J,M,A sont affectées des valeurs j,m,a.

• un *rectangle doublé* (désigne un bloc de calcul).
Le module correspondant sera traité en *sous-programme*.

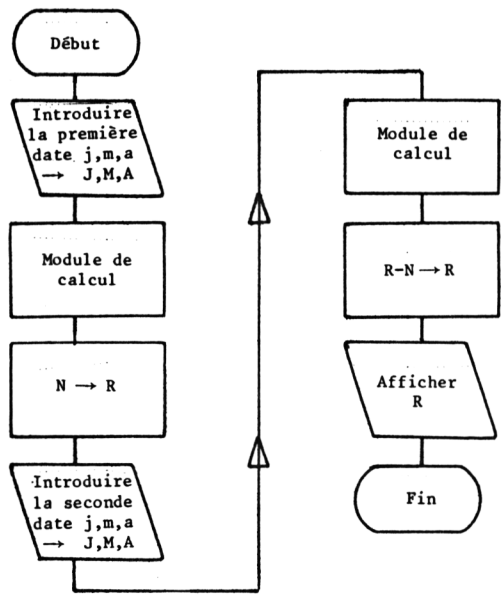
• un *rectangle* (désigne un calcul, ou de façon générale un traitement) contenant :

"N → R"

ce qui signifie que la valeur affectée à la variable N est affectée à R, ou plus simplement :

le contenu de la case mémoire N est mis dans la case mémoire R.

• et ainsi de suite.



D. PROGRAMMATION DE LA LOGIQUE DE DEROULEMENT

En langage BASIC, ceci se traduit par :

```
10 INPUT J,M,A
20 GOSUB 100
30 R=N
40 INPUT J,M,A
50 GOSUB 100
60 R=R-N
70 PRINT R
80 END
```

L'exécution commence en ligne 10 :

• En rencontrant l'instruction "*INPUT* J,M,A", l'ordinateur affiche "?" et attend trois nombres séparés par des virgules.

Une fois ces nombres tapés au clavier, leurs valeurs sont affectées aux variables numériques J,M,A (c'est-à-dire sont mises dans les cases mémoires notées J,M,A).

Si les entrées sont incorrectes (variables non numériques, moins ou plus de trois nombres), l'ordinateur affiche un message d'erreur et réexécute la ligne 10.

• En ligne 20, l'instruction "*GOSUB* 100" signifie que l'ordinateur va continuer l'exécution à la ligne 100 jusqu'à rencontrer l'instruction "*RETURN*", il reviendra alors à la ligne 30.

Vous remarquerez que j'ai réservé les lignes 100 et suivantes pour le module de calcul.

• En ligne 30, l'instruction "*R=N*" signifie que le contenu de la case mémoire notée N est mis dans la case mémoire notée R c'est-à-dire que la valeur affectée à la variable N est affectée à la variable R.

Il ne faut pas considérer ce symbole "=" comme un symbole d'égalité mais comme un *symbole d'affectation*.

Cela se voit encore mieux à la ligne 60 où l'instruction "R=R-N" signifie que le contenu de N est retranché au contenu de R et mis dans R.

Finalement, vous pouvez constater que ce programme transcrit fidèlement l'organigramme précédent.

Vous remarquerez également que les lignes 10-20 et 40-50 sont identiques, il est donc possible d'en économiser une en repoussant l'instruction "INPUT J,M,A" à la ligne 100 au début du module de calcul.

Cette remarque aurait pu être faite au moment de l'écriture de l'organigramme.

E. LE MODULE DE CALCUL

• Le seul problème délicat présenté par l'algorithme de calcul est de tenir compte des 29 Février des années bissextiles.

Je laisse au lecteur le soin de se persuader que le plus simple est de faire les calculs en comptant un mois de Février de 28,25 jours, le résultat final étant tronqué.

• Le calcul est alors :

$$365{,}25 \times (A-1901)$$

auquel on ajoute, selon le mois M, le nombre A(M) suivant :

M	1	2	3	4	5	6	7	8	9	10	11	12
A(M)	0	31	59,25	90,25	120,25	151,25	181,25	212,25	243,25	273,25	304,25	334,25

puis le nombre J.

Enfin, on prend la partie entière du résultat.

• J'ai donc besoin de constituer le tableau A( ), ce qui sera fait une seule fois en début de programme par 12 instructions d'affectation "A(1)=0" ... etc ... (ce qui peut-être réduit en utilisant les instructions READ et DATA de versions plus évoluées de BASIC).

• Une fois ceci fait le module est :

```
100 N=365,25*(A-1901)+A(M)+J
110 N=INT(N)
120 RETURN
```

La fonction *INT* (ligne 110) ne commande le calcul de la partie entière au sens mathématique que pour les arguments positifs.

Par exemple :

INT(187,5) est 187

mais

INT(-187,5) est -187 et non -188.

Ceci n'a aucune importance dans le cas de notre programme mais il faut y faire attention dans d'autres cas.

2. PROGRAMME DEFINITIF ET TEST DE CE PROGRAMME

A. LISTE DU PROGRAMME

Les lignes 10 à 20, 150 à 170, 240 à 260 ont été placées dans le but de rendre le programme plus lisible.

Les informaticiens disent que le programme est *documenté*.

L'ordinateur ne tient pas compte de ces lignes car elles commencent par "!".

```
 10 !      Initialisation
 20 !    ****************
 30 A(1)=0
 40 A(2)=31
 50 A(3)=59.25
 60 A(4)=90.25
 70 A(5)=120.25
 80 A(6)=151.25
 90 A(7)=181.25
100 A(8)=212.25
110 A(9)=243.25
120 A(10)=273.25
130 A(11)=304.25
140 A(12)=334.25
150 !
```

```
160 !      Traitement
170 !    ************
180 GOSUB 240
190 R=N
200 GOSUB 240
210 R=R-N
220 PRINT R
230 END
240 !
250 !    Sous-programme
260 !   ****************
270 INPUT J,M,A
280 N=365.25*(A-1901)+A(M)+J
290 N=INT(N)
300 RETURN
```

B. TEST DU PROGRAMME

Avant d'utiliser un programme - même simple - il est nécessaire de l'essayer sur des calculs faits à l'avance.

Ici, il est naturel d'essayer des calculs dont le résultat nous est évident :

31, 12, 1980 et 15, 11, 1980

par exemple.

Cela permet de déceler des erreurs ridicules comme celle que j'ai corrigée avant de vous offrir ce programme :

J'avais tapé : 290 N=INT(M) au lieu de : 290 N=INT(N).

Après, il est bon de faire un calcul plus compliqué pour lequel on utilise une calculatrice ordinaire, par exemple :

15, 05, 1905 et 28, 12, 1951

car il met en jeu des 29 Février.

Une fois le programme corrigé - un informaticien dirait débugé - vous pouvez l'utiliser (on en verra des applications en astronomie au chapitre 8).

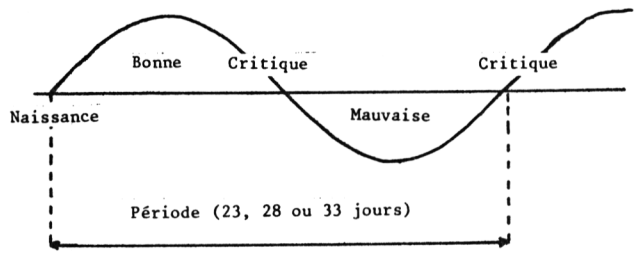
3. CALCUL DE BIORYTHMES

A. LA THEORIE DES BIORYTHMES

La théorie des biorythmes prétend que les êtres humains sont soumis à trois cycles périodiques :

- un cycle de 23 jours concernant la vitalité physique,
- un cycle de 28 jours concernant la sensibilité,
- un cycle de 33 jours concernant les facultés intellectuelles.

Ces cycles auraient pour origine le moment de la naissance et seraient invariables, ils sont représentés par :



Pour chacun de ces cycles, une journée serait bonne, mauvaise ou critique suivant le dessin précédent.

B. ANALYSE DU PROBLEME

Il est facile de modifier le programme précédent pour obtenir ces renseignements :

En effet, je peux calculer mon âge en jours soit n. Le reste de la division de n par 33 permets de me situer dans mon cycle intellectuel.

C. CALCUL DU RESTE D'UNE DIVISION

Tous les ordinateurs possèdent un programme de division, par $10 \div 3$, il donne : 3.333333333. En nombre entier, le quotient est 3 et le reste 1, c'est-à-dire $E(10 \div 3)$ et $10 - 3 \times E(10 \div 3)$.

Cette formule est générale, car si je me donne deux nombres entiers a et b, la division s'écrit :

$$a = bq + r \quad \text{avec} \quad 0 \leqslant r < b$$

donc $$q = \frac{a}{b} - \frac{r}{b}.$$

D'où, d'après la double inégalité vérifiée par r :

$$\frac{a}{b} - 1 < q \leqslant \frac{a}{b}$$

et donc $$q \leqslant \frac{a}{b} < q + 1$$

q est donc le plus grand entier inférieur à $\frac{a}{b}$: c'est la partie entière de $\frac{a}{b}$.

$r = a - bq$, c'est-à-dire : $r = a - bE\left(\frac{a}{b}\right)$

ce qui est bien la formule annoncée.

L'instruction informatique correspondante est :

```
R=A-B*INT(A/B)
```

Certains ordinateurs possèdent cette fonction, elle se note alors RMD, d'où l'instruction

```
R=RMD(A,B).
```

D. COMPLEMENT AU PROGRAMME

Pour calculer mes capacités intellectuelles, je complète donc le programme précédent par :

```
215 R=R-33*INT(R/33)
```

J'en déduis immédiatement ma position dans le cycle.

Vous remarquerez au passage pourquoi la numérotation initiale se fait de 10 en 10 : il est facile alors de réparer un oubli !

E. UTILISATION

Mon essai est concluant : je suis dans de bonnes dispositions intellectuelles (le résultat est 15).

Un doute me prend pourtant devant cet oracle : si mon horloge était un peu en avance ?

Hélas, pour un cycle de 33,01 jours, je suis dans la période fatidique (le résultat est 18).

Donc, même si cette théorie était vraie, il faudrait à nos horloges internes une fidélité dont ne sont pas dignes les meilleurs quartz pour nous donner des renseignements prévisionnels.

4. EXERCICES

Pour des dates postérieures au 15 Octobre 1582, programmez le calcul :

1. du nombre de jours entre deux dates.
2. du jour de la semaine en fonction de la date.

3. Dans l'acte V de Cyrano de Bergerac, dont l'action se situe en Septembre 1655, Edmond Rostand écrit :

"Et samedi vingt six, une heure avant dîné
Monsieur de Bergerac est mort assassiné".

Qu'en pensez-vous ?

4. Quel rapport voyez-vous entre l'assassinat de Henri III, la mort de Louis XV et la prise de la Bastille ?

5. Anne est née le mercredi 29 Février 1984, combien de fois pourra-t-elle fêter son anniversaire un mercredi si elle vit 78 ans ?

6. Reprenez les programmes précédents en utilisant le calendrier musulman moderne.
Faites un programme de conversion entre ce calendrier et le calendrier grégorien.

CHAPITRE 2

LES CALCULS ARITHMÉTIQUES SUR LES NOMBRES RATIONNELS

Le résultat désiré est d'effectuer des calculs du type :

$$\frac{3}{4}\times\frac{8}{9}-\frac{4}{5}+\frac{17}{7}\times\frac{14}{3}$$

de façon exacte, rapidement et sans erreur.

Ce qui, en effectuant les calculs à la main, est moins simple qu'il ne paraît. Par contre, les ordinateurs sont très bien adaptés à ce genre de problèmes. Mais pour cela il faut avant tout le définir clairement.

1. ANALYSE GENERALE

A. ANALYSE

- Mémorisation d'un nombre rationnel.

Il nécessite deux variables numériques (voir le paragraphe 1.B du chapitre 1) pour être mémorisé.

Ce qui me limite aux nombres $\frac{p}{q}$ tels que :

$$|p|<10^{11} \quad \text{et} \quad |q|<10^{11}.$$

Dans le cas - le plus courant - d'un ordinateur mémorisant les nombres avec 10 chiffres significatifs.

- Les calculs peuvent être effectués en chaîne donc il me faut :

- deux variables numériques P,Q pour y garder le résultat en cours.
- deux variables numériques R,S pour contenir le nouvel opérande.
- une variable alphanumérique C$ (une *variable alphanumérique* est une case

mémoire destinée à contenir des suites de signes quelconques à l'exception du signe " car il est utilisé pour les délimiter. L'ordinateur distingue les deux types de variables grâce au signe $ - dollars - qui termine les noms des variables alphanumériques) pour contenir le code de l'opération à effectuer :

"+" pour faire une addition.

"*" pour faire une multiplication.

J'y ajoute le code de fin de calcul :

"=" pour afficher le résultat.

- Le coeur du programme sera donc constitué par deux *modules d'opérations* comportant en entrée les nombres $\frac{p}{q}$, $\frac{r}{s}$ et en sortie le résultat $\frac{p}{q}$ (remis dans la zone mémoire du résultat en cours).

- Après chaque opération, il est nécessaire de simplifier le résultat d'où un *module de simplification* de $\frac{p}{q}$.

B. LOGIQUE DE DEROULEMENT DU PROGRAMME

A ce stage, la logique de déroulement du programme est dégagée.

L'organigramme ci-après se lit comme celui du paragraphe 1.C du chapitre 1, nous rencontrons cependant ici un symbole nouveau :

- un *losange* (désigne un test) contenant :

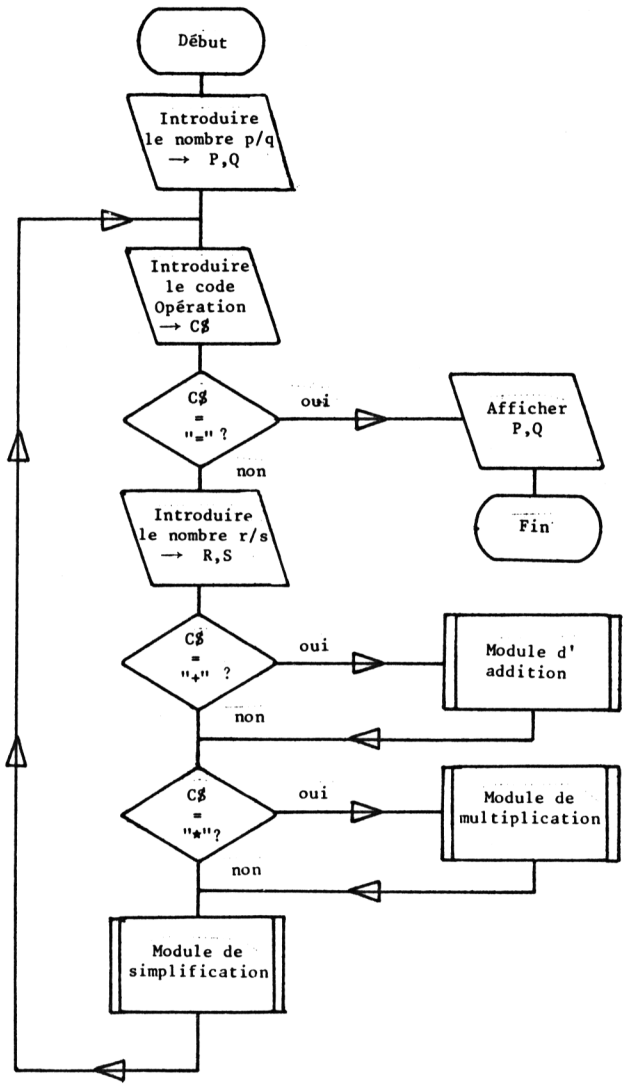
" C$="="? "

ce qui signifie que l'ordinateur teste si le contenu de C$ est "=", puis continue le traitement en suivant la flèche surmontée d'un oui si la réponse est oui, d'un non si la réponse est non.

Après avoir écrit un organigramme tel que celui-ci, *il est nécessaire de vérifier* que la suite d'opérations qu'il décrit résout bien le problème posé.

Pour cela, il suffit de le suivre en tenant le rôle de l'ordinateur. Ici, il est facile de voir que la première boucle est correcte. Le retour (flèche montante) s'opère au bon niveau car à ce moment, le premier résultat se trouve en P,Q. Les boucles suivantes sont identiques à la première, la sortie est assurée par le test C$="=".

L'organigramme est donc correct.


Début
Introduire le nombre p/q → P,Q
Introduire le code Opération → C$
C$ = "=" ?
oui
Afficher P,Q
Fin
non
Introduire le nombre r/s → R,S
C$ = "+" ?
oui
Module d' addition
non
C$ = "*"?
oui
Module de multiplication
non
Module de simplification

C. PROGRAMMATION DE LA LOGIQUE DE DEROULEMENT

En langage BASIC, ceci se traduit par :

```
10 INPUT P,Q
20 INPUT C$
30 IF C$="=" THEN PRINT P,Q : END
40 INPUT R,S
50 IF C$="+" THEN GOSUB 100
60 IF C$="*" THEN GOSUB 200
70 GOSUB 300
80 GOTO 20
```

Les instructions des lignes 10, 20, 40 et 70 ont déjà été étudiées au paragraphe 1.

- En ligne 30, l'instruction :

"IF C$="=" THEN PRINT P,Q : END"

signifie que l'ordinateur teste si le contenu de la case mémoire C$ est "=",

si oui il exécute l'instruction "PRINT P,Q"

c'est-à-dire qu'il affiche les contenus de P et Q, puis l'instruction "END" qui se passe de commentaires,

si non il passe à la ligne suivante : la ligne 40 ici.

- En ligne 80, l'instruction "GOTO 20" signifie que l'ordinateur va continuer l'exécution en ligne 20.

- Aux lignes 50, 60 et 70, vous remarquerez que j'ai réservé les lignes 100 à 199 pour le module d'addition, les lignes 200 à 299 pour le module de multiplication, les lignes 300 à 399 pour le module de simplification.

Finalement, vous pouvez constater que ce programme transcrit fidèlement l'organigramme précédent.

2. LES MODULES DE PROGRAMME

A. LES MODULES D'ADDITION ET DE MULTIPLICATION

- L'analyse du problème se résume aux formules :

$$\frac{p}{q}+\frac{r}{s}=\frac{ps+qr}{qs}$$

$$\frac{p}{q}\times\frac{r}{s}=\frac{pr}{qs}$$

Les résultats devant être affectés à P et Q.

• D'où les modules :

```
100 P=P*S+Q*R
110 Q=Q*S
120 RETURN

200 P=P*R
210 Q=Q*S
220 RETURN
```

• Comme au paragraphe 1.D du chapitre 1, le symbole "=" ne doit pas être pris pour un symbole d'égalité.

L'instruction de la ligne 100 signifie que l'ordinateur :

- multiplie les contenus des cases mémoires P et S.
- multiplie les contenus des cases mémoires Q et R.
- additionne les deux résultats précédents.
- affecte le résultat final à P.

B. LE MODULE DE SIMPLIFICATION

• Pour simplifier une fraction $\frac{p}{q}$, je divise p et q par leur p.g.c.d (plus grand commun diviseur) d. La façon la plus simple de le calculer est l'algorithme d'Euclide (mathématicien grec, IV°-III° siècle avant Jésus-Christ).

Par exemple, pour calculer le p.g.c.d. de 169 et 65.
Je divise d'abord 169 par 65 :

$$169 = 2 \times 65 + 39$$

et je remarque qu'un nombre divise 169 et 65 si et seulement si il divise 65 et 39
Je recommence alors sur 65 et 39 :

$$65 = 39 + 26$$

Un nombre divise 65 et 39 si et seulement si il divise 39 et 26

$$39 = 26 + 13$$

Un nombre divise 39 et 26 si et seulement si il divise 26 et 13

$$26 = 2 \times 13$$

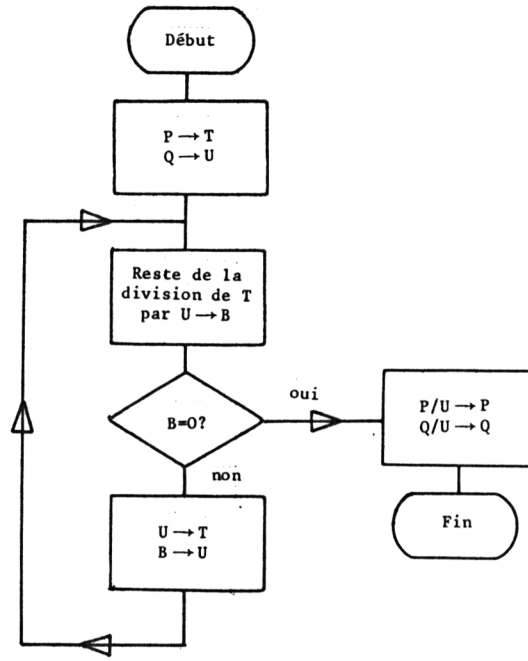
Donc un nombre divise 169 et 65 si et seulement si il divise 13, le p.g.c.d. de 169 et 65 est donc 13.

• Plus généralement, pour déterminer le p.g.c.d. de p et q, je divise p par q :

$$p = aq + b$$

où a et b sont des entiers et : $0 \leqslant b < q$.
Un nombre divise p et q si et seulement si il divise q et b.
Je recommence avec q et b qui prennent donc les rôles de p et q respectivement.
La suite des valeurs de q est strictement décroissante ($b < q$) donc après un nombre fini d'opérations $b = 0$ et la valeur finale de q est le p.g.c.d. cherché.

• L'organigramme est donc simple. La seule "finesse" étant de ne pas oublier de sauvegarder les contenus de P et Q avant l'opération, c'est-à-dire que l'algorithme s'opère sur des mémoires T et U :



- D'où le programme BASIC :

```
300 T=P
310 U=Q
320 B=T-U*INT(T/U)
330 IF B=0 THEN 370
340 T=U
350 U=B
360 GOTO 320
370 P=P/U
380 Q=Q/U
390 RETURN
```

l'instruction de la ligne 320 a été étudiée au paragraphe 3.C du chapitre 1.

3. UTILISATION ET AMELIORATION

A. UTILISATION

- Pour calculer $\frac{3}{4}\times\frac{8}{9}-\frac{4}{5}$, la liste des opérations à effectuer est :

Commandes	Affichage
RUN	?
3,4	?
*	?
8,9	?
+	?
-4,5	?
=	-2 15

Chaque commande étant suivie de la pression de la touche d'enregistrement (dont le nom varie suivant les ordinateurs : retour chariot sur les ancêtres (CR), ENDLINE, ENTER ou EXE de nos jours.

- Ce programme comporte un danger, en effet si un résultat intermédiaire comporte plus de 12 chiffres, alors les chiffres excédentaires sont perdus et le résultat final est donc faux.

Sur la plupart des ordinateurs, cela provoque un affichage en notation scientifique ce qui permet de le déceler.

Par mesure de sécurité, il est également possible de tester chaque résultat intermédiaire : je ne l'ai pas fait car dans la pratique courante les dépassements de capacité ne se produisent pas.

B. AMELIORATION

Ce programme n'est pas très pratique car il ne permet pas de garder les résultats intermédiaires, de même les opérations : - et ÷ peuvent être utiles.

Voici un programme mieux adapté que je vous laisse déchiffrer :

```
 10 ! Calcul arithmetique sur Q
 20 ! ************************
 30 !
 40 ! Introduction du premier
        nombre:P/Q
 50 !
 60 INPUT P,Q
 70 !
 80 ! Introduction du code
        operation C$
 90 !
100 ! =        : Visualisation du
             resultat en cours
110 ! G        : Conservation du
             resultat en cours
120 ! +,-,*,/  : Operations
130 !
140 INPUT C$
150 IF C$="=" THEN PRINT P,Q @ G
    OTO 140
160 IF C$="G" THEN G=P @ H=Q @ G
    OTO 60
170 !
180 ! Introduction du second
    nombre:R/S (si S=0 reprise
    du   resultat conserve)
190 !
200 INPUT R,S
210 IF S=0 THEN R=G @ S=H
220 !         Calcul
230 IF C$="+" THEN GOSUB 310
240 IF C$="-" THEN GOSUB 300
250 IF C$="*" THEN GOSUB 360
260 IF C$="/" THEN GOSUB 330
270 GOSUB 390
280 GOTO 140
290 !    Modules de calcul
300 R=-R !       Soustraction
310 P=P*S+Q*R ! Addition
320 GOTO 370
330 B=R !        Division
340 R=S
350 S=B
360 P=P*R !      Multiplication
370 Q=Q*S !      Calcul du
    denominateur (commun aux
    quatre operations)
380 RETURN
390 T=P
400 ! Module de simplification
410 U=Q
420 B=T-U*INT(T/U)
430 IF B=0 THEN 470
440 T=U
450 U=B
460 GOTO 420
470 P=P/U
480 Q=Q/U
490 RETURN
```

4. INVARIANT DE BOUCLE

Quand on réalise une boucle telle que celle utilisée pour transcrire l'algorithme d'Euclide (lignes 420 à 460), deux problèmes se posent :

. la sortie de boucle (ligne 430) s'effectue-t-elle après un nombre fini d'itérations ?

. à la sortie de la boucle, le résultat est-il bien celui que l'on désirait ?

Pour résoudre ces deux problèmes, une bonne méthode est de construire la boucle en utilisant une propriété invariante au niveau de la sortie de boucle (ici au moment de l'exécution de la ligne 430, les diviseurs des contenus de P et Q sont ceux des contenus de U et T) telle que le test de fin de boucle implique le résultat (ici, si B = 0 alors le contenu de U divise celui de T donc les diviseurs des contenus de P et Q sont ceux du contenu de U, c'est-à-dire qu'il est le p.g.c.d. cherché). D'autre part B = 0 après un nombre fini d'itérations d'après le raisonnement du paragraphe 2.B.

Cette méthode que nous utilisons dans tous les cas "épineux" est appelée méthode de l'invariant de boucle.

5. EXERCICES

Modifiez le programme du paragraphe 3.B. de façon que :

1. Les calculs s'effectuent suivant la notation polonaise inverse (avec ou sans pile opérationnelle).

2. L'ordinateur effectue le calcul des puissances entières (carré, cube, ... etc ...).

3. Plusieurs mémoires soient utilisables (au lieu d'une).

4. L'ordinateur signale les dépassements de capacité de calcul.

5. Programmez les calculs arithmétiques dans un corps quadratique $\mathbb{Q}(\sqrt{d})$ où d est un entier non carré (éventuellement négatif) :

Si δ est une racine carrée de d (c'est-à-dire un nombre tel que $\delta^2 = d$), $\mathbb{Q}(\sqrt{d})$ est l'ensemble des nombres de la forme :

$a + b\delta$ où a et b sont des nombres rationnels.

Il est facile de montrer que $\mathbb{Q}(\sqrt{d})$ est un corps, car :

$$(a_1+b_1\delta) + (a_2+b_2\delta) = (a_1+a_2) + (b_1+b_2)\delta$$

$$(a_1+b_1\delta)\ (a_2+b_2\delta) = (a_1a_2+b_1b_2d) + (a_1b_2+a_2b_1)\delta$$

$a + b\delta = 0$ si et seulement si $a = b = 0$, et si $a + b\delta \neq 0$:

$$\frac{1}{a+b\delta} = \frac{a-b\delta}{a^2-b^2d}$$

6. Une chaîne A$ étant donnée, écrivez un programme l'imprimant dans l'ordre inverse (PARIS devient SIRAP).

Pour cela, vous utiliserez un "pointeur" I et l'invariant de boucle suivant : "les éléments à partir du Ième sont imprimés dans l'ordre inverse".

CHAPITRE 3

CALCULS SUR LES POLYNÔMES

Je laisserai en exercices les calculs les plus faciles : addition et multiplication, pour étudier la *programmation de la division euclidienne*.

Celle-ci a un grand intérêt notamment pour les deux questions liées :

- *la factorisation des polynômes.*
- *la recherche des zéros d'un polynôme.*

(voir la méthode de Bairstow au chapitre 9 de la quatrième partie).

Dans ces deux cas, on se limite à diviser par des polynômes irréductibles et unitaires sur $\mathbb{R}$, c'est-à-dire de la forme :

$$x + a \quad \text{ou} \quad x^2 + px + q.$$

Au paragraphe 3, je traiterai également le problème du *calcul des valeurs d'un polynôme*.

1. DIVISION PAR x + a.

A. ANALYSE DU PROBLEME

La division s'effectue de la manière suivante :

$$\begin{array}{l|l} x^3 + 3x^2 + x - 2 & x - 1 \\ \cline{2-2} x^3 - x^2 & x^2 + 4x + 5 \\ \cline{1-1} \quad 4x^2 + x - 2 & \\ \quad 4x^2 - 4x & \\ \cline{1-1} \quad\quad 5x - 2 & \\ \quad\quad 5x - 5 & \\ \cline{1-1} \quad\quad\quad 3 & \end{array}$$

La première remarque est que les coefficients s'obtiennent de proche en proche suivant les puissances décroissantes, donc il est préférable d'écrire les polynômes suivant les puissances décroissantes :

$\sum_{i=o}^{n} a_i x^{n-i}$ pour le diviseur,

$\sum_{i=o}^{n} b_i x^{n-i}$ pour le quotient.

De façon immédiate :

$$b_0 = 0 \quad b_1 = a_0 \quad b_2 = a_1 - ab_1$$

(correspond à $3 - (-1)(1)$ de mon exemple).

De façon plus générale, il apparaît que :

$$b_i = a_{i-1} - ab_{i-1}$$

le reste étant alors b_{n+1}.

Il est facile de le vérifier par le calcul :

$$\left(\sum_{i=o}^{n} b_i x^{n-i}\right)(x+a) + b_{n+1} = b_0 x^{n+1} + \sum_{i=o}^{n} (b_{i+1} + ab_i) x^{n-i}$$

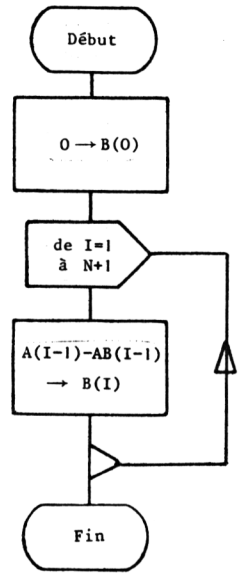
par identification :

$$b_0 = 0 \quad \text{et} \quad a_i = b_{i+1} + ab_i \text{ pour tout } i = 0, 1, \ldots, n.$$

B. PROGRAMMATION

Je laisse de côté le problème de l'introduction des données et de leur éventuelle correction (le programme du paragraphe 1.6 résoud cependant le premier, la résolution du second est un exercice facile).

L'organigramme suivant ne représente donc que le module de calcul proprement dit :



Cet organigramme se lit comme ceux du paragraphe 1.C du chapitre 1 et des paragraphes 1.B et 2.B du chapitre 2, cependant nous rencontrons ici deux symboles nouveaux :

- un *pentagone* (désigne le début d'une boucle à compteur) contenant

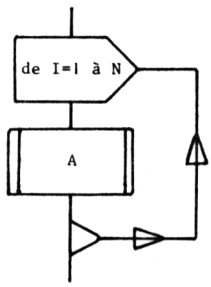
"de I=1 à N+1"

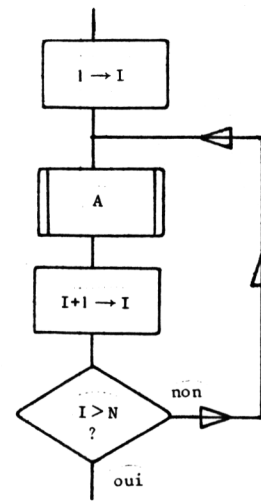
et :

- un *triangle* (désigne la fin d'une boucle à compteur).

Ce qui signifie que l'ordinateur exécute les instructions figurant entre ces deux symboles N+1 fois en affectant successivement à I les valeurs 1,2,...,N+1.

Les deux organigrammes suivant sont donc équivalents : (A désigne un module quelconque).





Le module de programme correspondant est :

```
10 B(0)=0
20 FOR I=1 TO N+1
30 B(I)=A(I-1)-A*B(I-1)
40 NEXT I
```

• En rencontrant l'instruction de la ligne 20 pour la première fois, l'ordinateur affecte 1 à la variable numérique I (voir le paragraphe 1.D du chapitre 1) puis passe à la ligne suivante.

• En rencontrant l'instruction de la ligne 40, il ajoute 1 à la variable I (on dit qu'il incrémente I de 1) et teste si $I \leqslant N+1$ si oui il revient à la ligne 20 sinon il passe à la ligne suivant la ligne 40.

• Donc, après chaque instruction NEXT, I est augmenté de 1. Sur la plupart des ordinateurs il est également possible d'utiliser un pas différent (positif ou négatif) grâce à l'instruction STEP.

Par exemple :

```
FOR I=10 TO 1 STEP-.5
```

signifie que I prendra successivement les valeurs 10 ; 9,5 ; 9 … etc … .
Cette propriété est particulièrement intéressante pour le tracé des courbes (voir le Tome 4).

C. PROGRAMME

```
10 ! Division d'un polynome par
20 !  un polynome de degre un
30 !  **********************
40 ! Introduction des donnees
50 !          Degre
60 PRINT "N=";
70 INPUT N
80 ! Coefficient du polynome
90 !     Σ A(I) X^N-I
100 FOR I=0 TO N
110 PRINT "A(";I;")=";
120 INPUT A(I)
130 NEXT I
140 ! Coefficient du diviseur
150 !         X + A
160 PRINT "A=";
170 INPUT A
180 !
190 !          Calcul
200 !
210 B(0)=0
220 FOR I=1 TO N+1
230 B(I)=A(I-1)-A*B(I-1)
240 NEXT I
250 !
260 !    Sortie des resultats
270 !
280 FOR I=1 TO N
290 PRINT "B(";I;")=";B(I)
300 NEXT I
310 PRINT "R=";B(N+1)
320 END
```

Le symbole ";" après une instruction PRINT (lignes 60, 110, 160, 290 et 310) signifie que l'ordinateur ne passe pas à la ligne pour une nouvelle écriture.

D. UTILISATION

Soit à factoriser le polynôme $P(x) = x^3 + 2x^2 + 2x + 1$.

Je remarque que $P(-1) = 0$, P est donc divisible par $x + 1$. Il est facile d'effectuer la division avec notre programme, la liste des opérations est (après RUN) :

Affichage	Introduction
N=?	3
A(0)=?	1
A(1)=?	2
A(2)=?	2
A(3)=?	1
A=?	1
B(1)=1	
B(2)=1	
B(3)=1	
R=0	

Donc

$$P(x) = (x+1)(x^2+x+1).$$

Ce qui prouve - en particulier - que P a un seul zéro :-1.

Dans le cas où les coefficients sont entiers le résultat est entier et donc exact (si toutefois les valeurs ne dépassent pas la capacité de l'ordinateur), sinon le résultat est approché. *Nous étudierons le problème de l'incertitude du résultat dans la troisième partie.*

2. DIVISION PAR $x^2 + px + q$.

A. PROGRAMME

Le programme est analogue au précédent. Le déchiffrer est un excellent exercice.

```
 10 ! Division d'un polynome par
 20 ! un polynome de degre deux
 30 !  **********************
 40 ! Introduction des donnees
 50 !          Degre
 60 PRINT "N=";
 70 INPUT N
 80 ! Coefficient du polynome
 90 !     Σ A(I) X^N-I
100 FOR I=0 TO N
110 PRINT "A(";I;")=";
120 INPUT A(I)
130 NEXT I
140 ! Coefficient du diviseur
150 !       X^2 + P X + Q
160 PRINT "P=";
170 INPUT P
180 PRINT "Q=";
190 INPUT Q
200 !
210 !          Calcul
220 !
230 B(0)=0
240 B(1)=0
250 FOR I=2 TO N+2
260 B(I)=A(I-2)-P*B(I-1)-Q*B(I-2
    )
270 NEXT I
280 !
290 !    Sortie des resultats
300 !
310 FOR I=2 TO N
320 PRINT "B(";I;")=";B(I)
330 NEXT I
340 ! Reste: R X + S
350 PRINT "R=";B(N+1)
360 PRINT "S=";B(N+2)+P*B(N+1)
370 END
```

B. UTILISATION

Soit à calculer : $x^5 + 6x^4 - 5x^3 + 2x^2 - x + 3$ pour $x = \frac{1+\sqrt{5}}{2}$.

Je remarque que ce nombre est racine du polynôme :

$$\left(x - \frac{1+\sqrt{5}}{2}\right)\left(x - \frac{1-\sqrt{5}}{2}\right) = \left(x - \frac{1}{2}\right)^2 - \frac{5}{4}$$

$$= x^2 - x - 1$$

Je divise $x^5 + 6x^4 - 5x^3 + 2x^2 - x + 3$ par $x^2 - x - 1$ à l'aide de mon programme, j'obtiens :

$$x^5 + 6x^4 - 5x^3 + 2x^2 - x + 3 = (x^3+7x^2+3x+12)(x^2-x-1) + 14x + 15$$

puisque $x^2 - x - 1 = 0$ pour $x = \frac{1+\sqrt{5}}{2}$, la valeur cherchée est :

$$14\,\frac{1+\sqrt{5}}{2} + 15 = 22 + 7\sqrt{5}.$$

3. CALCUL DES VALEURS D'UN POLYNÔME

A. PRESENTATION DES ALGORITHMES

Soit par exemple :

$$P(x) = 5x^5 + 4x^4 + 3x^3 + 2x^2 + 2x + 2.$$

Il s'agit de calculer la valeur de P pour des valeurs particulières de x : 2, 3, -1, π par exemple.

• Une première méthode - la plus immédiate - consiste à calculer les puissances de x de façon itérative :

$$\begin{aligned} &x \\ &x^2 = x \cdot x \\ &x^3 = x \cdot x^2 \\ &x^4 = x \cdot x^3 \\ &x^5 = x \cdot x^4 \end{aligned}$$

ce qui fait 4 multiplications.

Puis de les combiner ce qui fait 5 multiplications et 5 additions donc 14 opérations en tout.

• Une autre méthode - due à Newton mais appelée algorithme de Hörner - est d'écrire P sous la forme :

$$P(x) = 2 + x(2 + x(2 + x(3 + x(4+x(5)))))$$

le calcul se fait en sens inverse :

$$\begin{aligned} b_0 &= 5 \\ b_1 &= 4 + xb_0 \\ b_2 &= 3 + xb_1 \\ b_3 &= 2 + xb_2 \\ b_4 &= 2 + xb_3 \\ P(x) = b_5 &= 2 + xb_4 \end{aligned}$$

ce qui fait 5 multiplications et 5 additions donc 10 opérations en tout.

• Dans le cas d'un polynôme de degré n, le premier algorithme nécessite 3n-1 opérations et le second 2n ce qui est une économie de 30%.

B. ANALYSE DE L'ALGORITHME DE HÖRNER

Ici encore nous avons intérêt à écrire P suivant les puissances décroissantes :

$$P(x) = \sum_{i=o}^{n} a_i x^{n-i}.$$

La suite $(b_i)_{0 \leqslant i \leqslant n}$ est définie par

$$b_0 = a_0 \quad \text{et} \quad b_i = b_{i-1} x + a_i \text{ pour tout } 1 \leqslant i \leqslant n$$

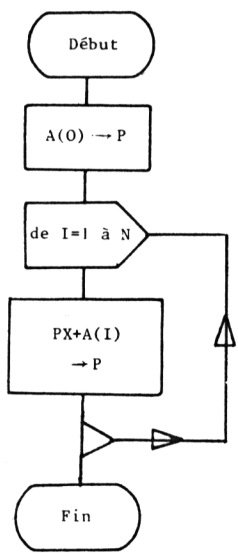
b_n est alors la valeur de P(x).

En effet, l'hypothèse :

$$b_i = \sum_{j=o}^{i} a_j x^{i-j} \text{ pour } 0 \leqslant i \leqslant n$$

se démontre très facilement par récurrence.

L'organigramme des calculs est donc immédiat :



A la fin, le résultat est affecté à la variable P.

C. PROGRAMME

```
10 ! Calcul des valeurs d'un
20 !        polynome
30 !      ***********
40 ! Introduction des donnees
50 !          Degre
60 PRINT "N=";
70 INPUT N
80 ! Coefficient du polynome
90 !    Σ A(I) X^N-I
100 FOR I=0 TO N
110 PRINT "A(";I;")=";
120 INPUT A(I)
130 NEXT I
140 !      Valeur de X
150 PRINT "X=";
160 INPUT X
170 !
180 !         Calcul
190 !
200 P=A(0)
210 FOR I=1 TO N
220 P=P*X+A(I)
230 NEXT I
240 !
250 !   Sortie du resultat
260 !
270 PRINT "P(";X;")=";P
280 END
```

D. UTILISATION

Si nous utilisons ce programme pour X et les coefficients du polynôme entiers (pas trop grands) le résultat est exact, sinon nous obtenons un résultat approché.

Le problème de l'incertitude de ce résultat sera étudié dans la troisième partie (chapitre 6, paragraphe 4.B.)

4. EXERCICES

Programmez les calculs :

1. de la somme et du produit de deux polynômes.

2. de la division suivant les puissances croissantes (le diviseur étant de degré un ou deux).

3. Appliquez les exercices 1 et 2 au calcul de développements limités et donc de limites.

En utilisant les modules des programmes des chapitres 2 et 3, programmez les calculs (exacts, bien entendu) :

4. de la somme et du produit de deux polynômes à coefficients rationnels.

5. de la division euclidienne d'un polynôme à coefficients rationnels par un polynôme de degré un ou deux à coefficients rationnels.

6. de la valeur en un point rationnel d'un polynôme à coefficient rationnel.

7. de développements limités à coefficients rationnels.

8. Reprenez les (ou des) calculs précédents dans le cas de coefficients dans un corps quadratique (voir exercice 5 du chapitre 2). En particulier, programmez des calculs tels que :

$$x^5 + 6x^4 - 5x^3 + 2x^2 - x + 3$$

pour $x = \frac{1+\sqrt{5}}{2}$.

9. Pouvez-vous étendre ces calculs à des polynômes à coefficients paramétriques ? Par exemple :

$$x^2 + ax + 1.$$

10. Programmez le calcul du p.g.c.d. de deux polynômes. De même que dans les exercices précédents, vous commencerez par des polynômes à coefficients réels, puis rationnels, puis … .

11. Programmez les calculs sur les fractions rationnelles à coefficients rationnels.

Etudiez l'utilisation de paramètres.

12. Peut-on programmer les calculs sur des expressions trigonométriques ? des expressions contenant des logarithmes, exponentielles … etc … ?

Etudiez ce type de problèmes.

Algorithmes de tri :

Un tableau de chaînes de caractères A$(.) étant donné, il s'agit de l'écrire dans l'ordre lexicographique (ordre utilisé dans les dictionnaires).

13. Ecrivez un programme permettant de trouver le plus petit élément du tableau A$(.). En permutant celui-ci avec le premier élément du tableau, déduisez-en un programme de tri. Vous préciserez les invariants de boucles utilisés. Voir également le tome 3.

14. *Tri par transposition.*

Le principe de cet algorithme est de comparer successivement les éléments consécutifs, si ils ne sont pas dans l'ordre, on les permute et on recommence aux précédents jusqu'à ce que le tableau soit trié.

Ecrivez un programme utilisant ce principe, vous préciserez les invariants de boucles utilisés.

15. *Tri par insertion.*

Il s'agit du tri que l'on utilise naturellement pour classer des noms écrits sur des feuilles séparées.

Pour le programmer, vous utiliserez un tableau de nombres contenant les indices du successeur de chaque élément du tableau et une variable contenant le premier indice.

Peut-on utiliser l'algorithme de recherche dichotomique comme sous-programme ? (voir l'exercice 10 du chapitre 7).

DEUXIEME PARTIE

ARITHMÉTIQUE
ET
APPLICATION À LA CRYPTOGRAPHIE

Dans la première partie, tous les algorithmes rencontrés sont de "bons" algorithmes dans le sens que leur exécution est rapide et n'augmente pas de façon considérable avec la taille des données.

Dans ces conditions, nous n'avons jamais eu de recherche à faire pour diminuer les temps d'exécution.

Il n'en est pas toujours ainsi comme le montre l'exemple du chapitre 4 :

la décomposition des nombres en facteurs premiers.

Cette difficulté même a une application importante en cryptographie : c'est le sujet du chapitre 5.

CHAPITRE 4

FACTORISATION D'UN NOMBRE

Le résultat désiré est d'effectuer des décompositions du type :

$$18819 = 3^3 \times 17 \times 41$$

où les nombres 3, 17 et 41 sont premiers, c'est-à-dire n'ont pas d'autres diviseurs qu'eux-mêmes et 1.

Une telle décomposition est unique, elle s'effectue en divisant par les nombres premiers successifs :

La première question à résoudre est donc de *tester la primarité d'un nombre*.

1. LES NOMBRES PREMIERS

A. LE CRIBLE D'ERATOSTHENE

Il est facile d'obtenir les nombres non premiers : ce sont les multiples des nombres supérieurs ou égaux à 2.

D'où la méthode illustrée par la figure suivante :

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

Je barre les multiples stricts de 2, puis de 3, de 5, etc. ... Les nombres restants sont les nombres premiers. L'algorithme est simple et rapide, son dé-

faut est d'exiger une mémoire considérable. Ainsi, pour obtenir les nombres premiers jusqu'à 1000, il faut utiliser un tableau de 1000 nombres !

La méthode est donc à rejeter.

B. TEST DE DIVISIBILITE

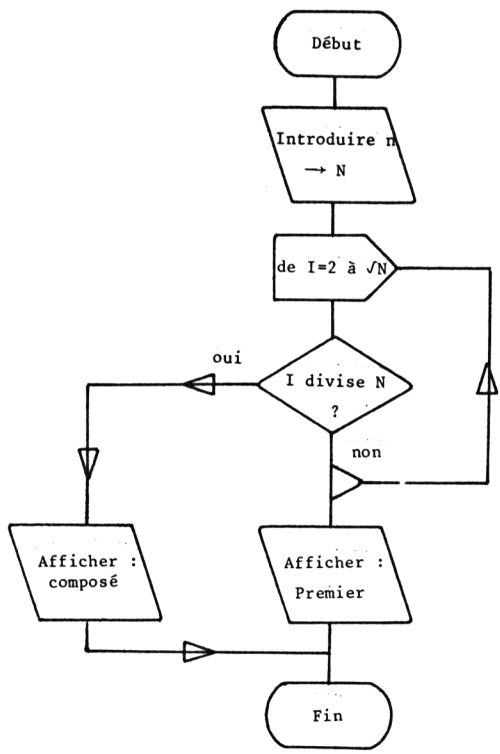
• Pour vérifier si un nombre n est premier, il suffit de tester si un des nombres entre 2 et n-1 le divise. Ce qui fait n-2 divisions.

On peut facilement réduire les calculs en remarquant que si :

$$n = a \cdot b$$

alors le plus petit des nombres a et b est inférieur ou égal à $\sqrt{n}$. Il reste donc $E(\sqrt{n})-1$ divisions.

• D'où l'organigramme des calculs :



• D'où le programme :

```
10 INPUT N
20 FOR I=2 TO SQR(N)
30 D=N-I*INT(N/I)
40 IF D=0 THEN 80
50 NEXT I
60 PRINT "Premier"
70 GOTO 90
80 PRINT "Composé"
90 END
```

à la ligne 20, la fonction SQR calcule la racine carrée, sur certains ordinateurs elle s'écrit $\surd$.

• Pour des ordres de grandeurs équivalents, le temps d'exécution est plus long pour un nombre premier :

c'est pour cela que je l'ai compté pour quelques nombres premiers.

n	1039	9967	52361	1247263	45236921	568945213	2147483647
t	1	2	4	24	142	403	990
$\surd$n	32	100	229	1117	6726	23853	46341
t/$\surd$n	0,03	0,02	0,02	0,02	0,02	0,02	0,02

le temps t étant compté en secondes et correspond à l'utilisation d'un ordinateur de table ; de façon approximative :

$$t = 0{,}02\surd n.$$

Il est prévisible que par ce procédé un nombre premier voisin de 10^{12} demandera un temps de 20 000 secondes, c'est-à-dire 5H30, pour être testé avec ce programme.

Il est donc intéressant d'essayer d'améliorer cet algorithme.

C. AMELIORATIONS

• Une première idée est de ne diviser que par 2 et les nombres impairs, les autres nombres n'étant pas premiers. Il est vraisemblable qu'alors le temps de calcul est divisé par deux.

Le programme devient :

```
 10 INPUT N
 20 D=N-2*INT(N/2)
 30 IF D=0 THEN 100
 40 FOR I=3 TO SQR(N) STEP 2
 50 D=N-I*INT(N/I)
 60 IF D=0 THEN 100
 70 NEXT I
 80 PRINT "Premier"
 90 GOTO 110
100 PRINT "Composé"
110 END
```

d'où les temps d'exécution :

n	1039	9967	52361	1247263	45236921	568945213	2147483647
t	0,5	1	2	12	71	252	495

Comme prévu, ils sont divisés par deux.

Remarquez que, du point de vue du temps d'exécution, les répétitions des lignes 20, 30 d'une part et 50, 60 d'autre part sont économiques : l'utilisation d'un sous-programme ferait perdre du temps.

- La méthode peut encore être améliorée car parmi les nombres :

$$6p,\ 6p+1,\ 6p+2,\ 6p+3,\ 6p+4,\ 6p+5$$

seuls 6p+1 et 6p+5 peuvent être premiers.

Donc, exceptés 2 et 3, les nombres premiers sont tous de la forme 6p+1 ou 6p+5.

Dans le programme, nous testons 2 et 3 puis i-1 et i+1 pour i variant de 6 en 6 de 6 à $\sqrt{n}+1$:

```
 10 !   Test de primarite
 20 !  ******************
 30 !     Introduction
 40 INPUT N
 50 !   Test du diviseur 2
 60 D=N-2*INT(N/2)
 70 IF D=0 THEN 210
 80 !   Test du diviseur 3
 90 D=N-3*INT(N/3)
100 IF D=0 THEN 210
110 ! Tests des diviseurs 6i-1
120 !   et 6i+1 jusqu'a n^.5
130 FOR I=6 TO SQR(N)+1 STEP 6
140 D=N-(I-1)*INT(N/(I-1))
150 IF D=0 THEN 210
160 D=N-(I+1)*INT(N/(I+1))
170 IF D=0 THEN 210
180 NEXT I
190 PRINT "Premier"
200 GOTO 220
210 PRINT "Compose"
220 END
```

d'où les temps d'exécution :

n	1039	9967	52361	1247263	45236921	568945213	2147483647
t	0,25	0,7	1,5	8	46	165	324

qui sont divisés par 3 par rapport aux premiers.

D. GENERALISATION

• L'idée peut évidemment se généraliser car si a est un nombre alors parmi les nombres :

$$ap + b \quad \text{où} \quad 1 \leqslant b \leqslant a - 1$$

seuls les nombres correspondants aux b premiers avec a peuvent être premiers.

Si $\varphi(a)$ est le nombre de nombres premiers avec a inférieurs à a, le nombre de divisions à effectuer est majoré par $\frac{\varphi(a)}{a}\sqrt{n}$. Il s'agit donc de choisir a tel que $\frac{\varphi(a)}{a}$ soit minimum.

• Si $p_1, p_2, \cdots, p_k$ sont les facteurs premiers de a, on montre (voir exercice 13) que :

$$\varphi(a) = a\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)$$

donc :

$$\frac{\varphi(a)}{a} = \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right).$$

Les nombres a les plus intéressants à considérer sont donc les produits des premiers nombres premiers :

a	2	2×3=6	2×3×5=30	2×3×5×7=210	2×3×5×7×11=2310
$\frac{\varphi(a)}{a}$	0,5	0,33	0,27	0,23	0,21

mais le gain est faible au-delà de 30, de plus il est compensé par une programmation plus longue (bien que simple).

J'ai ainsi obtenu un temps de 243 secondes pour tester la primarité de 2147483647 en adoptant a = 30.

E. UTILISATION

• Une des premières utilisations est la création d'une *table de nombres premiers*.

Voici un extrait de ma table :

256019	256021	256031	256033
256049	256057	256079	256093
256117	256121	256129	256133
256147	256163	256169	256181
256187	256189	256199	256211
256219	256279	256301	256307
256313	256337	256349	256363
256369	256391	256393	256423
256441	256469	256471	256483
256489	256493	256499	256517
256541	256561	256567	256577
256579	256589	256603	256609
256639	256643	256651	256661
256687	256699	256721	256723
256757	256771	256799	256801
256813	256831	256873	256877
256889	256901	256903	256931
256939	256957	256967	256981

- Ceci permet également d'étudier la *répartition des nombres premiers.*

Si je note $\pi(n)$ le nombre de nombres premiers inférieurs ou égaux à n, une légère modification du programme du paragraphe 1.C permet de calculer $\pi(n)$ pour les valeurs successives de n.

Le tableau suivant résume mes tentatives pour déterminer une formule approchée de $\pi(n)$:

n	$\pi(n)$	$\frac{n}{\pi(n)}$	Logn	$\frac{\pi(n)\text{Logn}}{n}$	$\frac{n}{\text{Logn}} + \frac{n}{(\text{Logn})^2}$	$\left[\pi(n) - \frac{n}{\text{Logn}} - \frac{n}{(\text{Logn})^2}\right]\frac{(\text{Logn})^3}{n}$
100	26	3,8	4,6	1,2	26	- 0,4
200	47	4,2	5,3	1,2	45	1,6
300	63	4,8	5,7	1,2	62	0,7
400	79	5,1	6,0	1,2	78	0,6
500	96	5,2	6,2	1,2	93	1,2
1000	169	5,9	6,9	1,2	166	1,1
2000	304	6,6	7,6	1,2	298	1,4
3000	431	7,0	8,0	1,1	422	1,6
4000	551	7,3	8,3	1,1	540	1,5
5000	670	7,5	8,5	1,1	656	1,7
10000	1230	8,1	9,2	1,1	1204	2,1
15000	1755	8,5	9,6	1,1	1722	1,9
20000	2263	8,8	9,9	1,1	2223	1,9

Dans un premier temps, un lien apparaît entre $\frac{n}{\pi(n)}$ et Logn. La formule que l'on peut espérer :

$$1,1\ \frac{n}{\text{Logn}} \leqslant \pi(n) \leqslant 1,2\ \frac{n}{\text{Logn}} \quad \text{pour} \quad 100 \leqslant n \leqslant 20000$$

n'est pas très précise.

Il est alors logique de chercher un terme complémentaire du type $K\ \frac{n}{(\text{Logn})^2}$ où K est une constante.

Il s'avère que :

$$\left|\pi(n) - \frac{n}{\text{Logn}} - \frac{n}{(\text{Logn})^2}\right| \leqslant 3\,\frac{n}{(\text{Logn})^3} \text{ pour tout } n \leqslant 20000$$

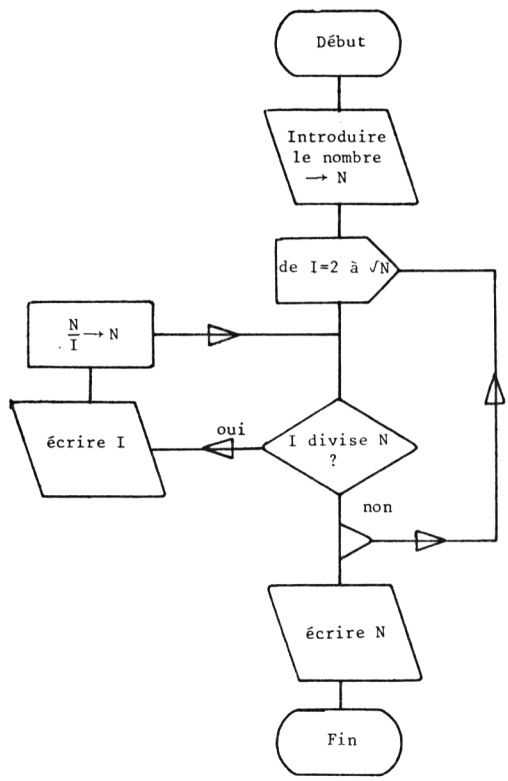
Comme je l'ai vérifié par un calcul exhaustif (et non seulement pour les valeurs du tableau).

Si vous êtes intéressé par une approche théorique de cette question, vous pouvez consulter le livre d'A. Blanchard : "Introduction à la théorie analytique des nombres premiers".

2. FACTORISATION D'UN NOMBRE

A. ADAPTATION DES PROGRAMMES PRECEDENTS

Prenons l'exemple du programme du paragraphe 1.B, si nous trouvons un diviseur I, il suffit de l'écrire et de continuer en remplaçant N par N/I suivant l'organigramme :



Evidemment, les temps de calcul seront plus faibles avec les améliorations du paragraphe 1.

Dans le paragraphe 2.B, je donne, sans commentaires, le programme correspondant à la recherche sur les suites 6n-1 et 6n+1.

B. PROGRAMME

Ce programme, que je vous laisse déchiffrer, est rapide pour les nombres ayant jusqu'à 9 chiffres

```
 10 ! Factorisation d'un nombre
 20 ! ************************
 30 !       Introduction
 40 INPUT N
 50 ! Diviseurs 2 et 3
 60 D=2 @ GOSUB 130
 70 D=3 @ GOSUB 130
 80 ! Diviseurs 6j+-1
 90 D=5 @ I=2 ! Increment
100 GOSUB 130
110 IF Q<=D THEN PRINT N @ END
120 D=D+I @ I=6-I @ GOTO 100
130 !  Sous-programme de test
140 !       d'un diviseur
150 IF N=1 THEN PRINT @ END
160 Q=INT(N/D) @ R=N-D*Q
170 IF R=0 THEN PRINT D;@ N=Q @
    GOTO 130
180 RETURN
```

Le programme de test de primarité du paragraphe 1.C peut être modifié dans le même esprit, comparez les temps d'exécution (le résultat est-il surprenant ?).

C. UTILISATION

(Résolution de quelques équations diophantiennes).

a. Soit à résoudre l'équation $x^2 - y^2 = 394616$ en nombres entiers.

Elle s'écrit : $(x+y)(x-y) = 394616$, donc $x+y$ et $x-y$ sont deux diviseurs complémentaires de 394616.

Il est donc nécessaire de chercher les diviseurs de 394616. Pour cela, je décompose 394616 en facteurs premiers à l'aide de mon programme :

$$394616 = 2^3 \times 107 \times 461$$

La décomposition : 2×197308 donne le système :

$$\begin{cases} x + y = 2 \\ x - y = 197308 \end{cases}$$

qui est équivalent à

$$\begin{cases} 2x = 197310 \\ 2y = -197306 \end{cases}$$

qui a une et une seule solution : $x = 98655$, $y = -98653$.

De même, une décomposition : $394616 = a \cdot b$ donne le système :

$$\begin{cases} x + y = a \\ x - y = b \end{cases}$$

qui est équivalent à :

$$\begin{cases} 2x = a + b \\ 2y = a - b \end{cases}$$

qui a une solution si et seulement si a et b sont de même parité donc tous les deux pairs puisque 394616 est pair.

D'où les décompositions licites (au signe près) :

$$2 \times 197308, \quad 4 \times 98654, \quad 214 \times 1844, \quad 428 \times 922.$$

Les solutions sont donc :

$x = \pm 98655$	$y = \pm 98653$
$x = \pm 49329$	$y = \pm 49325$
$x = \pm 1029$	$y = \pm 815$
$x = \pm 675$	$y = \pm 247$

b. Soit à résoudre l'équation : $x^4 - 21x^3 - 432x^2 - 364x + 816 = 0$ en nombres entiers.

Si x est solution : $816 = x(-x^3 + 21x^2 + 432x + 364)$ donc x divise en 816. Il s'agit donc de chercher les diviseurs de 816 :

$$816 = 2^4 \times 3 \times 17$$

Les diviseurs de 816 sont donc : $(-1)^\alpha \times 2^\beta \times 3^\gamma \times 17^\delta$ où : $0 \leqslant \alpha \leqslant 1$, $0 \leqslant \beta \leqslant 4$, $0 \leqslant \gamma \leqslant 1$, $0 \leqslant \delta \leqslant 1$. Donc il y a 40 essais à faire : il vaut mieux les faire par programme en utilisant le module de calcul du paragraphe 3.C. du chapitre 3 en sous-programme. Je trouve : - 12, - 2, 1 et 34.

c. Soit à résoudre l'équation : $34x^5 - 195x^4 + 53x^3 + 440x^2 - 421x + 105 = 0$ en nombres rationnels.

Si $x = \frac{p}{q}$ (où p et q sont des entiers premiers entre eux) est solution :

$$34p^5 - 195p^4q + 53p^3q^2 + 440p^2q^3 - 421pq^4 + 105q^5 = 0,$$

donc, de même que précèdemment, p divise 105 et q 34.

$$105 = 3 \times 5 \times 7 \quad \text{et} \quad 34 = 2 \times 17,$$

donc, il reste 64 essais à faire ce que l'on fait par programme (mais, attention à ne pas faire des calculs approchés !).

3. EXERCICES

1. Programmez les améliorations du test de la primarité d'un nombre proposées au paragraphe 1.D. ($a = 30$, $a = 210$).
Etudiez le gain de temps ainsi obtenu.

2. Reprenez l'exercice 1 pour la factorisation d'un nombre.
Suivant le matériel dont vous disposez, étudiez l'intérêt de l'utilisation d'une table de nombres premiers (sur disque ou cartouche magnétique).
Programmez une méthode mixte. Comparez les temps d'exécution.

3. n étant un entier, soit p_n le $n^{\text{ième}}$ nombre premier. Etudiez expérimentalement la fonction $n \longrightarrow p_n$, essayez d'en déterminer une valeur approchée.

4. Améliorez l'approximation de $\pi(n)$ obtenue au paragraphe 1.E pour $n \leqslant 20\,000$.
Etudiez $\pi(n)$ sur un intervalle plus grand.

5. On appelle nombres premiers jumeaux deux nombres premiers dont la différence est égale à 2.
Etudiez leur fréquence expérimentalement.

6. Si a est un entier, étudiez expérimentalement la distribution des nombres premiers entre les différentes suites $(an+b)_{n \in \mathbb{N}}$ où b est un entier.

7. On appelle nombre parfait tout nombre entier égal à la somme de ses diviseurs stricts (par exemple : $6 = 3 + 2 + 1$).
Etudiez expérimentalement les nombres parfaits.
Montrez que $2^{n-1}(2^n-1)$ est un nombre parfait si 2^n-1 est premier.
Pensez-vous que cette formule donne tous les nombres parfaits ?

8. Résolvez les équations suivantes en nombres entiers :

$$x^2 - y^2 = 1681 \qquad x^2 - 4y^2 = 25682 \qquad x^2 - y^2 = 254.$$

Plus généralement, déterminez la condition de compatibilité de l'équation : $x^2 - y^2 = A$ où A est un nombre entier (c'est-à-dire la condition sur A pour que l'équation ait au moins une solution).

9. Résolvez les équations suivantes en nombres entiers :

$$x^2 + y^2 = 126 \qquad x^2 + 9y^2 = 421 \qquad x^2 + y^2 = 377.$$

Pour cela vous ferez tous les essais qui s'imposent (12 essais pour la première, par exemple).
En écrivant : $x^2 + y^2 = (x+iy)(x-iy)$, remarquez que le problème peut également se poser en termes de décomposition en facteurs premiers dans l'anneau Z(i) (c'est-à-dire des nombres de la forme $a+bi$ où a et b sont des entiers).
Si vous êtes intéressés par cette question, un chapitre du livre d'A. Blanchard cité au paragraphe 1.E. lui est consacré.

10. Résolvez les équations suivantes en nombres rationnels :

$$298x^7 - 2127x^4 + 29x^3 - 78x^2 + 2130x - 319 = 0$$
$$3150x^3 - 7515x^2 + 4401x - 324 = 0$$

11. Ecrivez un programme résolvant les équations algébriques en nombres entiers.

12. Ecrivez un programme résolvant les équations algébriques en nombres rationnels.

13. n étant un entier, on note $\varphi(n)$ le nombre de nombres premiers avec n inférieurs à n. $\varphi(n)$ est appelé l'indicateur d'Euler de n.

a. Si n et m sont premiers entre eux, montrez que :

$$\varphi(mn) = \varphi(m)\varphi(n).$$

b. Calculez $\varphi(p^{\alpha})$ pour p premier et α entier supérieur ou égal à 1.

c. Si $n = p_1^{\alpha_1} p_2^{\alpha_2} \dots p_k^{\alpha_k}$ est la décomposition de n en facteurs premiers, montrez que :

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right).$$

14. Soit n un nombre premier et a un nombre entier vérifiant : $1 \leqslant a \leqslant n-1$.

a. Montrez que les restes de a, 2a, … , (n-1)a dans la division par n sont tous différents et non nuls.

Déduisez-en que le reste de a^{n-1} dans la division par n est 1. Ce résultat est appelé le théorème de Fermat.

b. Montrez que n = 561 vérifie également ce résultat.

c. Peut-on utiliser ce qui précède comme test de primarité ?

On appelle nombres de Carmichael les nombres non premiers vérifiant le théorème de Fermat. Nous verrons en exercice 10 du chapitre 5 un test de primarité fondé sur ce théorème.

d. Si vous avez étudié l'ensemble Z/nZ, simplifiez vos démonstrations en l'utilisant.

15. Trouvez tous les entiers (a,b,c,d,e,f,g,h) inférieurs à 20 tels que :

$$\begin{cases} a + b + c + d = e + f + g + h \\ a^2 + b^2 + c^2 + d^2 = e^2 + f^2 + g^2 + h^2 \\ a^3 + b^3 + c^3 + d^3 = e^3 + f^3 + g^3 + h^3 \end{cases}$$

16. Trouvez les nombres égaux à la somme des cubes de leurs chiffres.

17. *Suite de Conway*

John Conway définit deux suites de nombres entiers (u_n) et (v_n) de la façon suivante :

Il pose $u_0 = 2$ puis, si u_n est défini, u_{n+1} est le produit de u_n par le premier nombre de la suite : $\frac{17}{91}, \frac{78}{85}, \frac{19}{51}, \frac{23}{38}, \frac{29}{33}, \frac{77}{29}, \frac{95}{23}, \frac{77}{19}, \frac{1}{17}, \frac{11}{13}, \frac{13}{11}, \frac{15}{14}, \frac{15}{2}$, 55 tel que le résultat soit entier.

Dans la suite (u_n) ainsi obtenue, il note 2^{v_0}, 2^{v_1}, … les puissances de 2 rencontrées.

Calculez les premiers termes de la suite (v_n). Que constatez-vous ?

Essayez de le démontrer.

(Vous remarquerez que les termes de la suite (u_n) s'écrivent sous la forme :
$2^a 3^b 5^c 7^d m$ où $m \in \{1, 11, 13, 17, 19, 23, 29\}$ et considérerez a+c et b+d).

CHAPITRE 5

APPLICATION À LA CRYPTOGRAPHIE

La difficulté même de trouver un algorithme rapide pour décomposer un grand nombre en facteurs premiers a donné l'idée de fonder une méthode de cryptographie sur cette propriété :

La clef de codage est un nombre $n = p_1 p_2$ (p_1 et p_2 étant premiers) et la clef de décodage le nombre $\varphi(n) = (p_1-1)(p_2-1)$ ce qui exige de connaître p_1 et p_2 et n'est donc pas facile même en connaissant n.

Avec les algorithmes que nous avons utilisés et le même type d'ordinateur, il faut environ un an si n a 17 chiffres.

Les meilleurs algorithmes connus ont un temps d'exécution de l'ordre de $\sqrt[7]{n}$ (au lieu de $\sqrt{n}$), cela donne quelques millions d'années pour un nombre de 400 chiffres même avec un ordinateur perfectionné (du type Cray 2).

1. PRESENTATION DE L'ALGORITHME

A. PROBLEME DE LA CRYPTOGRAPHIE

L'agent Z9 du SZRD de Bordurie rend compte à ses chefs des premiers résultats de sa mission en Syldavie. Pour cela, il leur transmet le *texte* :

T = "RAS Z9"

Pour éviter toute interception, il code T c'est-à-dire qu'il transforme T suivant une loi f, il obtient alors un nouveau texte :

M = f(T) que l'on appellera le *message*

f a donc le sens mathématique d'une fonction.

Ici :

$$f(T) = \text{"180119002639"}.$$

(Comme chacun le sait, les Bordures ne sont pas très inventifs et vous trouverez facilement la fonction f utilisée).

Au centre du SZRD, les agents décodent le message c'est-à-dire utilisent une fonction g telle que :

$$g(M) = T.$$

Donc le problème de la cryptographie est de trouver deux fonctions f et g définies sur l'ensemble des textes possibles telles que :

$$g \circ f = id$$

id désignant l'application identité.

Bien entendu, g doit être le plus difficile possible à trouver. L'idéal étant que g soit également difficile à trouver même si f est connue : ainsi, même Z9 ne pourrait révèler la clef du décodage Bordure.

B. L'ALGORITHME

Nous avons vu avec la méthode Bordure que nous pouvons nous limiter à des nombres que nous décomposerons en groupes de cinq chiffres dans l'exemple qui suit.

On choisit deux nombres premiers assez grands p_1 et p_2 — 277 et 349 pour fixer les idées — et on pose : $n = p_1 p_2$ — 96673. Alors l'indicateur d'Euler de n est : $\varphi(n) = (p_1-1)(p_2-1)$ — 96048 — comme nous l'avons vu au chapitre 4. On choisit alors deux nombres α et β tels que le reste de la division de $\alpha\beta$ par $\varphi(n)$ est 1 :

$$\alpha = 5 \quad \text{et} \quad \beta = 57629 \quad \text{conviennent.}$$

(La façon dont j'obtiens α et β sera étudiée au paragraphe 3.A).

Si $t \in [0,n-1]$ — $[0,96672]$ — alors f(t) est le reste de t^{α} — t^5 — dans la division par n — 96673 — et g(t) le reste de t^{β} — t^{57629}.

On démontre (voir l'exercice 6) qu'alors :

$$g \circ f = id.$$

Vous remarquerez que j'ai choisi α petit (c'est même le plus petit possible) de façon que le codage soit plus facile que le décodage.

2. UN PROGRAMME DE CRYPTOGRAPHIE

A. CODAGE

Le problème essentiel est de calculer la puissance cinquième d'un nombre de façon exacte. Pour cela, nous pouvons faire quatre multiplications successives.

Ensuite, nous calculons le reste du résultat dans la division par 96673.

L'inconvénient est que si nous opérons de cette manière, nous dépassons largement les capacités de la machine car t^5 peut avoir 25 chiffres. Il est cependant facile de contourner ce problème en prenant systématiquement le reste des résultats des résultats intermédiaires - procédé que nous justifierons au paragraphe 2.C.

D'où le programme :

```
10 !   Programme de codage
20 !  ********************
30 !      Introduction
40 INPUT T
50 !     Initialisation
60 M=T
70 !    Boucle de calcul
80 FOR I=1 TO 4
90 M=M*T
100 M=M-96673*INT(M/96673)
110 NEXT I
120 !  Sortie du resultat
130 PRINT M
140 END
```

Nous désirons coder le texte :

T = 180119002639

que nous décomposons en trois groupes de cinq chiffres :

f(18011) = 30213
f(90026) = 31579
f(39000) = 77701

d'où f(T) = 302133157977701.

B. DECODAGE

• En principe le programme est analogue mais ici l'exposant est 57629 ce qui fait 57628 multiplications. Nous pouvons cependant calculer t^{57629} plus rapidement en remarquant qu'on obtient rapidement de grandes puissances de t par l'algorithme

$$t \cdot t = t^2$$
$$t^2 \cdot t^2 = t^4$$
$$t^4 \cdot t^4 = t^8$$
$$t^8 \cdot t^8 = t^{16}$$

et ainsi de suite les exposants 2^n.

Or tout nombre est une somme de puissance de 2 : c'est l'écriture binaire d'un nombre.

Par ce procédé, le nombre d'opérations est de l'ordre de 2n, n étant tel que :

$$2^n \leqslant 57629 < 2^{n+1}, \text{ c'est-à-dire } 30.$$

Le gain de temps est considérable.

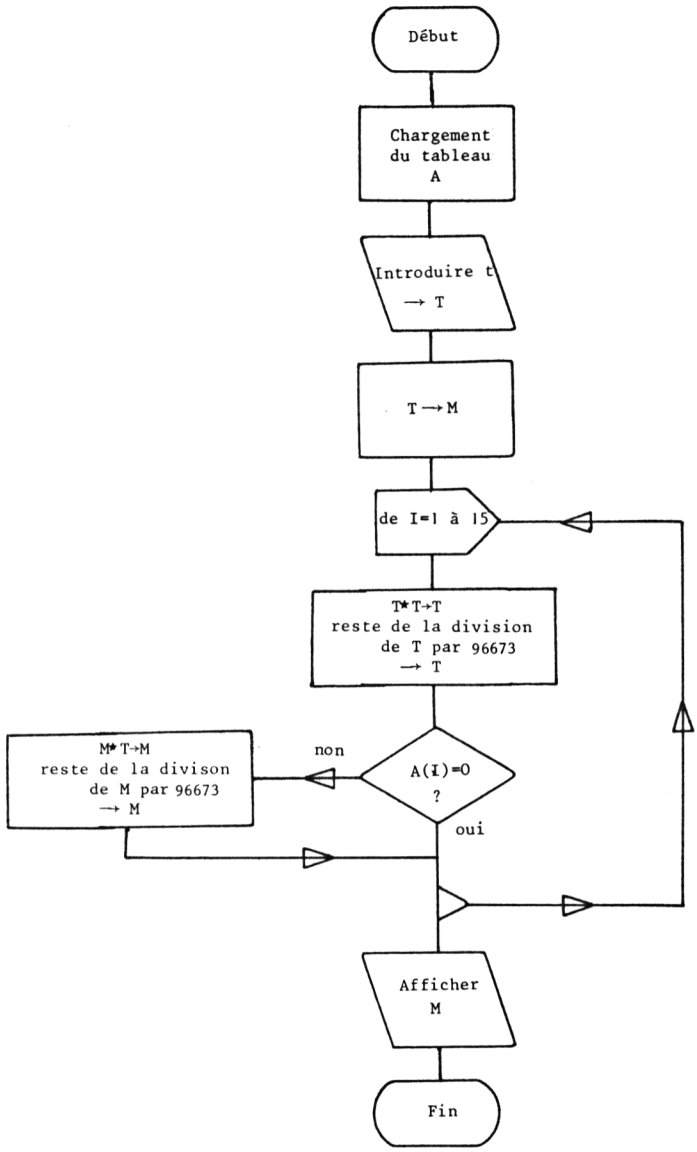
- Pour décomposer 57629 en binaire, il suffit d'utiliser le programme:

```
10 !   Ecriture d'un nombre
20 !        en base deux
30 ! ************************
40 INPUT N
50 ! Q=quotient de n par 2
60 Q=INT(N/2)
70 ! R=reste de n par 2
80 R=N-2*Q
90 PRINT R;
100 ! La boucle s'acheve quand
110 ! le quotient est nul.
120 IF Q=0 THEN END
130 N=Q
140 GOTO 60
```

Le résultat devant alors être lu de gauche à droite. Je trouve 1110000100011101

- Cet exposant est stocké dans un tableau A(I), A(I) étant le chiffre à la $I^{\text{ième}}$ position de l'écriture binaire de 57629 : si A(I) = 1, $t^{(2^I)}$ doit être multiplié à m sinon non, d'où le programme et l'organigramme de calcul :

```
10 !  Programme de decodage
20 ! ************************
30 !
40 !  Chargement du tableau A
50 !
60 FOR I=1 TO 15
70 READ A(I)
80 NEXT I
90 DATA 0,1,1,1,0,0,0,1,0,0,0,0
   ,1,1,1
100 !
110 !      Introduction
120 INPUT T
130 !   Initialisation de M
140 M=T
150 !
160 ! Calcul de T puissance 2^
170 !    pour i de 1 a 15
180 FOR I=1 TO 15
190 T=T*T
200 T=T-96673*INT(T/96673)
210 IF A(I)=0 THEN 260
220 ! Multiplication de T
230 ! puissance 2^i a M
240 M=M*T
250 M=M-96673*INT(M/96673)
260 NEXT I
270 !
280 !   Sortie du resultat
290 PRINT M
300 END
```



- Nous constatons bien que :

$$g(30213) = 18011$$
$$g(31579) = 90026$$
$$g(77701) = 39000$$

C. LES MATHEMATIQUES SOUS JACENTES

Dans l'exemple précédent, j'ai défini deux fonctions inverses l'une de l'autre de [0,96672] dans lui-même. Leur étude est simplifiée en remarquant que cet ensemble - que l'on note Z/96673 - jouit de propriétés analogues à Z, l'ensemble des nombres entiers relatifs.

Pour cela, on définit la somme et le produit de deux nombres de Z/96673 en remplaçant les résultats usuels par leurs restes dans la division par 96673, ainsi :

$$54401 + 85203 = 42931$$

et :

$$28902 \times 35204 = 79356.$$

Ces opérations ont les propriétés usuelles de l'addition et de la multiplication dans Z, autrement dit : Z/96673 est un anneau (voir l'exercice 7 pour une démonstration de ce résultat). Ceci justifie les procédés de calcul du paragraphe 2.A car l'opération $t \longrightarrow t^5$ s'y effectue dans Z/96673 et non dans Z. De plus chaque nombre a de Z/96673 représente tous les entiers dont le reste dans la division par 96673 est a, ainsi l'égalité :

$$96673 = 0 \text{ est vraie dans } Z/96673.$$

On l'écrit également $96673 \equiv 0 \pmod{96673}$ ce qui se lit 96673 est congru à 0 modulo 96673.

3. PROBLEME DU DECODAGE

Les services de contrespionnage syldaves ont découvert que Z9 utilise la méthode suivante pour coder ses messages :

Il code son message en chiffres par la numérotation alphabétique, il obtient un nombre qu'il découpe en groupes de six chiffres en commençant par la gauche. Puis il élève chaque groupe à la puissance 7 et prend le reste du résultat dans la division par 937169.

Ils ont intercepté le message :

917063414124019270093776210133872294137460882399263351001661117940.

Quelle est sa signification ?

A. DECODAGE

L'algorithme utilisé est du type étudié au paragraphe 1, il s'agit donc de trouver $\varphi(937169)$ puis β tel que le reste de la division de 7β par $\varphi(937169)$ soit 1.

Je décompose 937169 en facteurs premiers grâce au programme du chapitre 4 :

$$937169 = 859 \times 1091$$

d'où

$$\varphi(937169) = 858 \times 1090 = 935220$$

Je cherche β tel que : $7\beta = 935220k + 1$ où k est un entier, ce qui est très simple à résoudre si l'on considère cette équation comme une équation en k; elle est alors équivalente à l'équation dans Z/7 :

$$6k + 1 = 0$$

car $935220 \equiv 6 \pmod 7$.

Cette équation admet comme solution : $k = 1$, d'où :

$$\beta = \frac{935221}{7} = 133603.$$

Il reste alors à écrire 133603 en binaire :

$$100000100111100011,$$

le programme de décodage du paragraphe 2.B. est alors facile à adapter. Je vous en laisse le soin.

B. CONCLUSION

Même quand la méthode de codage est connue, le décodage nécessite la factorisation d'un nombre : une telle opération est actuellement impossible en un temps raisonnable si ce nombre atteint 400 chiffres par exemple.

Cette méthode est particulièrement intéressante pour la protection des fichiers informatiques (ou tout autre protection d'ailleurs) :

l'accès étant conditionné par le décodage d'un texte aléatoire préalablement codé par l'ordinateur (qui ignore lui-même la méthode de décodage).

Dans un autre domaine, cette méthode peut servir de signature électronique : l'expéditeur étant le seul possesseur de sa méthode de codage, son utilisation l'identifie ainsi.

Les services secrets des différents pays l'utiliseront-ils ? Mon métier n'étant pas d'espionner les espions, je ne puis le dire.

4. AMELIORATION

Pour utiliser des nombres plus grands, il est nécessaire de pouvoir effectuer des sommes et des produits exacts de nombres ayant plus de chiffres. Par exemple, avec un ordinateur dont la précision est de 10 chiffres, seuls les produits de deux nombres d'au plus 5 chiffres sont exacts : ceci

explique mon choix de $n \leqslant 10^5$ pour le premier exemple (le second exigeant en fait ce qui suit). Pour utiliser n de l'ordre de 10^{10}, il faut donc effectuer les produits en *double précision*.

A. PRODUIT EN DOUBLE PRECISION

Le but est donc d'effectuer des calculs du type :

$$1023456789 \times 2453670189$$

de façon exacte.

Etant limité a priori à des produits de nombres de cinq chiffres, une idée est d'écrire les nombres en base 10^5, ainsi :

$$1023456789 = 10234 \times 10^5 + 56789$$
$$2453670189 = 24536 \times 10^5 + 70189.$$

On effectue ensuite les produits :

10234×24536	10^{10}
10234×70189	10^5
56789×24536	10^5
56789×70189	1

Le résultat est écrit en base 10^{10} : $a\,10^{10} + b$. Pour répartir les produits effectués entre a et b, il faut écrire 10234×70189 et 56789×24536 - les coefficients de 10^5 - en base 10^5 puis régler le problème de la retenue dans le cas ou : $b \geqslant 10^{10}$ après ces opérations.

Le programme suivant résoud ces problèmes :

```
 10 !  Multiplication en double
 20 !         precision.
 30 !        ***********
 40 ! Introduction des donnees.
 50 ! Introduction du premier
 60 ! nombre: A(0)+A(1)*10^5.
 70 GOSUB 310
 80 A(1)=A @ A(0)=B
 90 ! Introduction du second
100 ! nombre: B(0)+B(1)*10^5.
110 GOSUB 310
120 B(1)=A @ B(0)=B
130 !         Calcul.
140 !        ********
150 ! Produits en colonnes.
160 C(0)=A(0)*B(0)
170 C(1)=A(1)*B(1)
180 ! Produits croises.
190 X=A(0)*B(1)
200 GOSUB 330
210 GOSUB 370
220 X=A(1)*B(0)
230 GOSUB 330
240 GOSUB 370
250 !   Sortie du resultat.
260 !  ********************
270 DISP C(1);C(0)
280 END
290 !      Sous-programmes.
300 !     *****************
310 ! Introduction d'un nombre.
320 INPUT X
330 ! Mise sous forme:A*10^5+B
340 A=INT(X/100000)
350 B=X-100000*A
360 RETURN
370 ! Addition des produits
380 !         croises.
390 C(1)=C(1)+A
400 X=C(0)+100000*B
410 ! Y-a-t-il une retenue?
420 IF X<10000000000 THEN 470
430 ! Retenue.
440 C(1)=C(1)+1
450 B=B-100000
460 X=C(0)+100000*B
470 C(0)=X
480 RETURN
```

Remarque : ce type de programme utilitaire est à rédiger en langage assembleur de préférence. Le temps gagné est alors très important.

B. AMELIORATION DU PROGRAMME

Je cherche deux nombres premiers voisins de 10^5 au moyen du programme du chapitre 4, soit par exemple :

$$99929 \quad \text{et} \quad 100043.$$

J'obtiens : $n = 99929 \times 100043 = 9997196947$ et $\varphi(n) = 99928 \times 100042 = 9996996976$ non divisible par 3 donc premier avec $\alpha = 3$.

D'après l'identité de Bachet (voir l'exercice 8 pour une démonstration de ce résultat), il existe deux nombres entiers u et v tels que :

$$\alpha u + \varphi(n)v = 1$$

c'est-à-dire qu'il existe bien β tel que :

$$\alpha\beta = 1 \text{ dans } Z/\varphi(n) \quad [\beta = u, \text{ bien entendu}].$$

Pour le trouver, j'écris :

$$3\beta = 1 + k\varphi(n) \text{ où } k \text{ est un entier.}$$

Et je remarque que k existe si et seulement si : $1 + k = 0$ dans $Z/3$ car $\varphi(n) \equiv 1 \pmod 3$. $k = 2$ est solution, d'où :

$$\beta = \frac{1 + 2\varphi(n)}{3} = 6664664651.$$

Le programme est alors facile à rédiger.

C. GENERALISATION

Dans tous mes exemples, j'ai choisi pour simplifier : $n = p_1p_2$ où p_1 et p_2 sont premiers distincts. En fait, la condition pour appliquer ce qui précède est que n soit sans facteur carré, c'est-à-dire :

$$n = p_1p_2 \cdots p_r \quad \text{où } p_1, p_2, \cdots, p_r$$

sont des nombres premiers deux à deux distincts.

Le choix de α premier avec $\varphi(n)$ assure alors l'existence de β tel que :

$$\alpha\beta = 1 \text{ dans } Z/\varphi(n).$$

Et les deux applications : $t \longrightarrow t^{\alpha}$ et $t \longrightarrow t^{\beta}$ de Z/n dans lui-même sont alors réciproques l'une de l'autre.

D. LE DERNIER PROBLEME

Il reste un dernier problème (mais de taille !) : comment trouver des nombres premiers très grands ? (de 100 chiffres par exemple). Pour cela, il n'est pas question d'appliquer l'algorithme - trop lent - du chapitre 4. On possède actuellement une méthode beaucoup plus rapide de test de primarité (en $K \operatorname{Log}^4 n$ au lieu de $K\sqrt{n}$), elle est fondée sur une amélioration du théorème de Fermat (voir l'exercice 14 du chapitre 4 et l'exercice 10).

Si vous êtes intéressé, vous consulterez avec profit l'article de G. Miller (Université de Berkeley) intitulé : L'hypothèse de Riemann et les tests de primarité, et, également, celui de R. Solovay et V. Strassen : Une méthode de Monte-Carlo rapide pour tester la primarité, améliorée (et rendu non probabiliste) par H. Cohen de l'Université de Bordeaux.

5. EXERCICES

1. Ecrivez le programme de codage proposé au paragraphe 4.B. Vous remarquerez que pour cela vous n'avez pas besoin de programmer une division en double précision.

2. Ecrivez un programme de protection des fichiers de votre ordinateur. Vous pouvez même y inclure une autodestruction des programmes et(ou) des fichiers !

3. Programmez les différents calculs arithmétiques en double - ou triple - précision.
Ces programmes sont très utiles surtout si vous pouvez les rédiger en langage assembleur, ils sont alors très rapides.

4. Programmez le calcul de très grandes puissances sur l'idée du paragraphe 2.B. Remarquez qu'une écriture préalable de l'exposant en binaire est inutile : elle peut être simultanée.

5. Programmez les calculs arithmétiques dans Z/n, du calcul de l'inverse quand celui-ci existe, ainsi que la résolution des équations algébriques dans Z/n.

6. Soit $n = p_1 p_2$ où p_1 et p_2 sont des nombres premiers distincts, k un entier tel que : $k \equiv 1 \pmod{\varphi(n)}$.

a. Montrez que : $k \equiv 1 \pmod{(p_1-1)}$ et $k \equiv 1 \pmod{(p_2-1)}$.

b. En utilisant le théorème de Fermat (voir l'exercice 14 du chapitre 4), montrez que p_1 et p_2 divisent $x^k - x$ pour tout x.

c. Déduisez-en que : $x^k \equiv x \pmod{n}$ pour tout x.

d. Montrez que si α et β sont tels que : $\alpha\beta \equiv 1 \pmod{\varphi(n)}$ alors les applications $t \longrightarrow t^\alpha$ et $t \longrightarrow t^\beta$ de Z/n dans lui-même sont réciproques l'une de l'autre.

e. Généralisez au cas où n est sans facteur carré.

7. n étant un entier, on définit sur Z la relation :

$$a \equiv b \pmod{n}$$

ce qui signifie que a - b est divisible par n.

a. Montrez que cette relation est une relation d'équivalence compatible avec l'addition et la multiplication dans Z, c'est-à-dire que :

- la relation est réflexive ($a \equiv a$ pour tout a), symétrique ($a \equiv b$ implique $b \equiv a$) et transitive ($a \equiv b$ et $b \equiv c$ implique $a \equiv c$).

- $a \equiv b$ implique $a + c \equiv b + c$ et $ac \equiv bc$ pour tout c.

b. Déduisez-en que Z/n muni de l'addition et de la multiplication (définies au paragraphe 2.C) est un anneau commutatif, c'est-à-dire qu'elles vérifient les règles de calcul usuelles sur Z :

$$a+b = b+a \qquad ab = ba$$

$$(a+b)+c = a+(b+c) \qquad a(bc) = (ab)c$$

$$a(b+c) = ab+ac$$

$$a+0 = a \qquad a \cdot 1 = a \qquad a \cdot 0 = 0$$

et, pour tout a, il existe un nombre que l'on note (-a) tel que : $a+(-a) = 0$.

c. Montrez que Z/6 n'est pas intègre c'est-à-dire qu'il existe deux nombres a et b non nuls tels que : $a \cdot b = 0$.

Remarquez qu'ainsi on ne peut pas simplifier une expression du type $ab = ac$ en $b = c$.

8. Soit a et b deux entiers premiers entre eux.
Soit d le plus petit entier strictement positif de la forme :

$$au+bv \quad \text{où} \quad u \in Z \quad \text{et} \quad v \in Z.$$

En divisant $au+bv$ par d, montrez que $au+bv$ est un multiple de d pour tout u et v.
Déduisez-en que : $d = 1$, c'est-à-dire qu'il existe u et v dans Z tels que :

$$au+bv = 1.$$

ce qui constitue l'identité de Bachet.

9. En utilisant l'identité de Bachet, montrez que si n est premier alors Z/n est un corps (c'est-à-dire que tout élément non nul y admet un inverse).

10. Programmez le calcul de 3^{n-1} modulo n pour n entier inférieur à 10^{10}.
D'après le théorème de Fermat (voir l'exercice 14 du chapitre 4), si n est premier alors le résultat est 1 mais la réciproque est fausse. L'expérience montre (pouvez-vous la faire ?) que les nombres inférieurs à 10^{10} composés et vérifiant $3^{n-1} \equiv 1 \pmod{n}$ ont un facteur premier inférieur à 1000 sauf le nombre 38347921.
Programmez ce test de primarité.

11. En écrivant les nombres en base 10^5, programmez l'addition de nombres à 20 "chiffres" (c'est-à-dire à 100 chiffres dans le système décimal).

Ecrivez un programme de multiplication d'un nombre de 20 chiffres par un nombre d'un chiffre puis déduisez-en un programme de multiplication de deux nombres.

Programmez également la soustraction et la division euclidienne.

TROISIEME PARTIE

LE CALCUL APPROCHÉ

Jusqu'à présent, nous n'avons effectué que des calculs exacts : le seul problème était celui de trouver un algorithme efficace.

Bien souvent, les calculs que nous pouvons faire ne sont qu'approchés, c'est-à-dire qu'au lieu de calculer une valeur x, nous en calculons une *valeur approchée* $\overline{x}$. L'*erreur* commise en assimilant x à $\overline{x}$ est :

$$x - \overline{x}.$$

Cette erreur est, par définition, inconnue car si elle était connue, elle n'existerait pas !

En général, nous pouvons connaître un majorant de sa valeur absolue, c'est-à-dire un nombre Δx tel que :

$$|x-\overline{x}| \leqslant \Delta x.$$

Δx est alors appelée l'*incertitude* (ou *incertitude absolue*) sur le résultat x.

Vous remarquerez aisément qu'un résultat approché n'a aucune signification si on ne connaît pas son incertitude.

Ceci constitue le problème majeur du calcul approché, malheureusement il n'est pas toujours résolu de manière satisfaisante.

Dans le chapitre 6, nous analysons les diverses origines des erreurs et nous voyons des premiers exemples de calculs d'incertitude.

CHAPITRE 6

LES ERREURS ET LEURS ORIGINES

Les erreurs sont de deux types :

- les *erreurs dues à la méthode* employée,
- les *erreurs dues aux limites de l'outil de calcul utilisé* (ou erreurs de calculs) ; par exemple, pour mon ordinateur :

$$\frac{1}{3} = 0{,}3333333333.$$

Remarquez qu'un autre ordinateur pourra utiliser 100 chiffres qu'il restera toujours une erreur. (Bien entendu, nous excluons ici le cas de calculs exacts sur Q, $Q(\sqrt{2})$, $Q(\pi)$, $Q(e)$, $Q(\pi,e)$, ... etc ... comme il a été vu au chapitre 2).

Ces erreurs de calcul peuvent avoir des répercussions plus ou moins grandes sur le résultat suivant l'algorithme utilisé, c'est le :

problème de la stabilité des algorithmes par rapport aux erreurs de calcul.

1. LES DIFFERENTES ORIGINES DES ERREURS

A. ETUDE D'UN EXEMPLE

Soit à calculer : $x = e^3 + 2e + 1$, e étant la base des logarithmes néperiens.

e n'étant pas connu de façon exacte, nous ne pouvons qu'en utiliser une valeur approchée $\overline{e}$ pour les calculs.

Nous calculons donc : $\overline{x} = \overline{e}^3 + 2\overline{e} + 1$, d'où une erreur de méthode :

$$x - \overline{x} = (e-\overline{e})(e^2 + e\overline{e} + \overline{e}^2 + 2).$$

Si l'incertitude sur e est $\Delta e = 10^{-9}$ (e est tronqué à dix chiffres), en majorant e et $\bar{e}$ par 3 :

$$|x - \bar{x}| \leqslant 29\Delta e,$$

d'où : $\Delta x = 29\Delta e$ convient.

Cette incertitude est appelée *incertitude due à la méthode* car elle ne tient pas compte du fait que le calcul de : $\bar{e}^3 + 2\bar{e} + 1$ se fait de façon approchée.

Quand l'ordinateur effectue les calculs en *virgule flottante* sur 10 chiffres, $\bar{e}^3$ est tronqué à 10^{-8}, il introduit donc une erreur supplémentaire d'au plus 10^{-8} de même pour $\bar{e}^3 + 2\bar{e}$ ce qui fait 2×10^{-8}.

D'où finalement : $\Delta x = 5 \times 10^{-8}$ ce qui n'empêche pas l'ordinateur d'afficher un résultat comportant 10 chiffres significatifs mais le dernier est douteux, donc : $x = 26{,}5221006$ à 10^{-7} près.

B. INCERTITUDE DUE A LA METHODE

Nous voyons donc qu'elle dépend essentiellement de la méthode utilisée. Voici un cas classique :

f étant une fonction de $\mathbb{R}$ dans $\mathbb{R}$, soit à calculer : $y = f(x)$ où : $x = \bar{x}$ avec une incertitude de Δx. La méthode consiste à calculer la valeur : $\bar{y} = f(\bar{x})$, bien entendu.

L'erreur de méthode est :

$$y - \bar{y} = f(x) - f(\bar{x}).$$

Si f est dérivable sur $[x - \Delta x,\ x + \Delta x]$ et si il existe un nombre $M > 0$ tel que :

$$|f'(t)| \leqslant M \text{ pour tout } t \in [x - \Delta x,\ x + \Delta x]$$

alors, l'inégalité des accroissements finis s'écrit :

$$|f(x) - f(\bar{x})| \leqslant M|x - \bar{x}|$$

et donc : $\Delta y = M\Delta x$ convient comme incertitude due à la méthode sur y.

Si $f'(x) \neq 0$, M et $|f'(\bar{x})|$ sont assez voisins, il est donc possible de choisir pour une évaluation rapide : $\Delta y = |f'(\bar{x})|\Delta x$ mais il ne faut pas oublier que ce calcul d'incertitude n'est alors qu'approximatif.

Ceci est encore applicable pour une fonction de plusieurs variables $y = f(x_1, x_2, \ldots, x_n)$, alors :

$$\Delta y = \sum_{i=1}^{n} \left| \frac{\partial f}{\partial x_i}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n) \right| \Delta x_i,$$

et dans le cas de la résolution d'une équation dont les coefficients sont des

valeurs approchées, la solution étant considérée comme une fonction implicite des coefficients (voir l'exercice 3 du chapitre 9).

C. INCERTITUDE DUE AUX CALCULS

Sur ordinateur, le calcul de $f(\bar{x})$ ne se fait en général pas exactement du fait du nombre de chiffres limité qu'il utilise.

Ces deux types d'incertitude dépendent de l'algorithme utilisé. Dans la suite du chapitre, je vais m'intéresser essentiellement aux erreurs de calcul et passer en revue les différents cas de perte de précision.

2. LES TRONCATURES ET LES ARRONDIS

Si un ordinateur utilise dix chiffres, il écrit les nombres sous la forme :

$$a \times 10^b$$

où a est un nombre de dix chiffres et b un nombre de deux chiffres.

Quand il effectue une opération arithmétique, tout se passe comme s'il effectuait le résultat exact puis tronquait a à dix chiffres (le premier n'étant pas un zéro).

Ainsi : $345678 \times 123609 = 4,27289119 \times 10^{10}$ car le résultat exact est 427289119 02.

De même : $685 + 10^{-8} = 685$.

Ce type d'opérations est appelée opération en *"virgule flottante"*.

Sur beaucoup d'ordinateurs, les résultats sont arrondis et non tronqués ce qui est préférable. Ainsi, 568941×234578 donne :

$1,334610419 \times 10^{10}$ si l'ordinateur arrondit,

$1,334610418 \times 10^{10}$ s'il tronque.

Cette différence - infime a priori - peut conduire à des résultats très différents.

3. CAS DE PERTE DE PRECISION

Vous allez voir sur les quelques exemples qui suivent que les erreurs dues aux calculs peuvent être si grandes que le résultat n'a alors plus aucune signification.

Le genre de "catastrophes" est bien résumé par les calculs suivants faits sur un ordinateur calculant sur dix chiffres :

$$1 + 10^{-10} - 1 = 0$$
$$1 - 1 + 10^{-10} = 10^{-10}.$$

C'est dire que l'addition en virgule flottante n'est ni associative, ni commutative !

A. DIFFERENCE EVANESCENTE

a. *Exemple*

Soit : $y = \sqrt{x+1} - \sqrt{x}$ pour $x \geqslant 0$,

pour $x = 10^9$, $y = 2 \times 10^{-5}$,

et pour $x = 10^{10}$, $y = 0$ bien entendu.

En écrivant : $y = \dfrac{1}{\sqrt{x+1} + \sqrt{x}}$, je trouve :

pour $x = 10^9$, $y = 1{,}5811388 \times 10^{5}$

pour $x = 10^{10}$, $y = 5{,}000000 \times 10^{-6}$.

b. *Analyse du phénomène*

En calculant : $y = \sqrt{x+1} - \sqrt{x}$ pour $x = 10^{10}$, l'ordinateur tronque $10^{10} + 1$ à 10^{10} donc $y = 0$, par contre $y = \dfrac{1}{\sqrt{x+1} + \sqrt{x}}$ devient $\dfrac{1}{2\sqrt{x}}$ d'où le résultat.

De même, pour $x = 10^9$, la précision est perdue.

B. ACCUMULATION DES ERREURS DE TRONCATURES

a. *Exemple*

On démontre en analyse que pour tout nombre x :

$$e^x = \sum_{p=o}^{+\infty} \frac{x^p}{p!}$$

c'est-à-dire : $e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots + \frac{x^p}{p!} + \ldots$.

Pour calculer e^x, il est alors logique de calculer pour un certain nombre n (que je déterminerais par la suite) :

$$s_n(x) = \sum_{p=o}^{n} \frac{x^p}{p!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots + \frac{x^n}{n!} .$$

L'erreur de méthode est alors :

$$r_n(x) = \sum_{p=n+1}^{+\infty} \frac{x^p}{p!} = \frac{x^{n+1}}{(n+1)!} + \frac{x^{n+2}}{(n+2)!} + \ldots ,$$

dont il s'agit de majorer la valeur absolue.

Une idée simple (qui est développée dans la première partie du tome 2) est d'étudier :

$$\frac{u_{p+1}(x)}{u_p(x)} \quad \text{où} \quad u_p(x) = \frac{x^p}{p!} .$$

$\frac{u_{p+1}(x)}{u_p(x)} = \frac{x}{p+1}$ donc : $\left|\frac{u_{p+1}(x)}{u_p(x)}\right| \leqslant \frac{|x|}{n}$ si $p \geqslant n$ donc, par récurrence :

$$|u_p(x)| \leqslant |u_n(x)| \left(\frac{|x|}{n}\right)^{p-n} \quad \text{pour tout } p \geqslant n.$$

D'où :

$$|r_n(x)| \leqslant |u_n(x)| \sum_{p=1}^{\infty} \left(\frac{|x|}{n}\right)^p \quad \text{pour tout } n > |x|$$

(condition pour que la série de droite converge).

Alors :

$$|r_n(x)| \leqslant |u_n(x)| \; \frac{|x|}{n} \frac{1}{1 - \frac{|x|}{n}} \quad \text{pour tout } n > |x|,$$

car, pour tout $|t| < 1$, $\sum_{n=o}^{\infty} t^n = \frac{1}{1-t}$ (voir l'exercice 11 pour une démonstration de ce résultat).

Donc :

$$|r_n(x)| \leqslant \frac{|x|^{n+1}}{n!(n - |x|)} \quad \text{pour tout } n > |x|, \text{ d'où l'incertitude :}$$

$$\Delta_n = \frac{|x|^{n+1}}{n!(n - |x|)} .$$

Prenons le cas du calcul de e^{-100} à 10^{-5} près par cette méthode, nous cherchons n tel que :

$$\Delta_n = \frac{10^{-5}}{2} .$$

$n = 280$ convient comme nous le montre le calcul itératif de Δ_n par le programme :

```
10 U=100
20 N=1
30 U=U*100/N
40 PRINT N;U/(N-100)
50 N=N+1
60 GOTO 30
```

[Bien entendu, ce programme entraîne un message d'erreur pour N = 100 mais cela n'est pas grave].

Le calcul de $s_{280}(-100)$ se fait alors suivant le programme :

```
10 S=1
20 U=1
30 FOR P=1 TO 280
40 U=-U*100/P
50 S=S+U
60 NEXT P
70 PRINT S
80 END
```

Le résultat est alors : $3{,}813089857 \times 10^{30}$ ce qui est sans aucune commune mesure avec le résultat : $e^{-100} = 0$ à 10^{-5} près.

b. *Analyse du phénomène*

Comme vous le remarquez, ici l'erreur de calcul est énorme : plus de 10^{30}. Voyons les calculs tels qu'ils sont effectués par l'ordinateur :

à la ligne 50, le contenu de U est ajouté à celui de S. Le contenu de U est d'abord 1 puis il augmente en valeur absolue jusqu'à P = 100 où il atteint $\frac{100^{100}}{(100)!}$ c'est-à-dire 10^{42} à peu près.

Du fait de la virgule flottante, les résultats sont donc arrondis à 10^{32} : l'erreur de 10^{30} est donc naturelle !

Remarquez qu'une meilleure méthode de calcul de e^{-100} est de calculer e^{100} par l'algorithme proposé puis son inverse.

C. INSTABILITE

a. *Exemple*

Je considère la suite $(u_n)_{n\geqslant 0}$ définie par :

$$\begin{cases} u_0 = \frac{1}{3} \\ \text{et} \\ u_{n+1} = 4u_n - 1 \text{ pour tout } n \geqslant 0 \end{cases}$$

et je me propose de calculer u_{100} pour me donner une idée de sa limite (éventuelle). Je fais le calcul suivant le programme :

```
10 U=1/3
20 FOR N=1 TO 100
30 U=4*U-1
40 NEXT N
50 PRINT U
60 END
```

Le résultat est approximativement -5×10^{49} ce qui me laisse penser que : $\lim_{n\to+\infty} u_n = -\infty$.

Or un peu plus d'attention me permet de remarquer que u_n est constamment égal à 1/3 !

b. *Analyse du phénomène*

La valeur affectée à U est 0,3333333333 et non $\frac{1}{3}$, soit $\frac{1}{3} - \varepsilon$ avec $0 < \varepsilon < 10^{-10}$.

Je note u'_0 cette valeur et je définis la suite $(u'_n)_{n\geqslant 0}$ par :

$$u'_{n+1} = 4u'_n - 1 \text{ pour tout } n \geqslant 0,$$

puis la suite $(v_n)_{n\geqslant 0}$ par : $v_n = u_n - u'_n$ pour tout $n \geqslant 0$.

Un raisonnement par récurrence élémentaire donne : $v_n = 4^n\varepsilon$ pour tout $n \geqslant 0$, donc : $\lim_{n\to+\infty} v_n = +\infty$ et donc : $\lim_{n\to+\infty} u'_n = -\infty$ ce qui explique le phénomène qualitativement.

Du point de vue quantitatif, il vient : $v_{100} = 4^{100}\varepsilon$ et :

$$\varepsilon = 0{,}33\ \ldots \times 10^{-10} = \frac{10^{-10}}{3}\ , \text{ donc : } v_{100} = \frac{4^{100} \times 10^{-10}}{3} = 5 \times 10^{49}$$

approximativement. Nous retrouvons la valeur précédente ! Le phénomène est donc complètement justifié.

Dans les cas de ce genre, nous disons que l'algorithme de calcul est instable car une erreur sur une valeur u_n a des conséquences graves sur les valeurs suivantes.

4. STABILITE

A. DEFINITION DE LA STABILITE D'UN ALGORITHME

Avec les exemples précédents, vous voyez que l'accumulation des erreurs de troncatures peut avoir des effets désastreux, dans ce cas on dit que *l'algorithme est instable.*

Si par contre, il est peu sensible à l'accumulation des erreurs de troncatures on dit qu'il est *stable*.

L'étude de la stabilité des algorithmes est souvent un problème difficile. Bien souvent on ne sait pas la faire de façon théorique et on se contente d'une étude expérimentale. Aussi, à la fin de certains calculs machines, on a simplement une croyance sur la précision du résultat. De façon générale, il est plus simple de vérifier le résultat obtenu a posteriori mais cela n'est pas toujours possible (c'est ce que nous ferons dans le cas de la résolution des équations).

B. STABILITE DE L'ALGORITHME DE HÖRNER

Soit à calculer :

$$P(x) = x^{11} + 9x^{10} - 7x^9 + x^8 + 3x^7 - 2x^6 + 5x^5 + 8x^4 + 2x^3 + x^2 - x + 4$$

pour : $x = 4{,}357801$.

Pour cela j'utilise le programme du paragraphe 3.D du chapitre 3. J'obtiens : $P(x) = 29241456{,}65$.

Etudions la propagation des erreurs sur cet exemple, la partie calcul de ce programme est :

```
200 P=A(0)
210 FOR I=1 TO N
220 P=P*X+A(I)
230 NEXT I
```

Je note Δ_i l'incertitude sur P et P_i la valeur de P à l'itération i.

$$A(0) \text{ étant exact, } \Delta_0 = 0.$$

L'exécution de la ligne 220 implique une erreur de méthode de : $|x|\Delta_i$ à laquelle il convient d'ajouter l'erreur de troncature : $|P_i|10^{-10}$ donc :

$$\Delta_{i+1} = |x|\Delta_i + |P_i|10^{-10}$$

Les suites $(\Delta_i)_{0 \leqslant i \leqslant n}$ et $(P_i)_{0 \leqslant i \leqslant n}$ sont donc définies par :

$$\begin{cases} \Delta_0 = 0 \qquad P_0 = 1 \\ P_{i+1} = xP_i + a_i \\ \Delta_{i+1} = |x|\Delta_i + |P_i|10^{-10} \text{ pour tout } 0 \leqslant i \leqslant n-1 \end{cases}$$

Le calcul de Δ_n peut donc être fait de façon simultanée au calcul de P_n, il suffit d'ajouter au programme les lignes :

```
205 D=0
225 D=ABS(X)*D+ABS(P)*1E-10
275 PRINT D
```

Ce qui donne : $\Delta = 4 \times 10^{-2}$, le résultat est donc :

$$P(x) = 29241456{,}6 \text{ à } 10^{-1} \text{ près.}$$

La précision est donc bonne ce qui est généralement le cas si le polynôme est de faible degré.

Remarquez que, bien entendu, D est tronqué à la ligne 225 d'où une légère minoration de l'erreur. On peut l'éviter en ajoutant un terme correctif qui provoque une majoration, par exemple :

```
226 D=D*(1+2E-9)
```

ce qui ne change d'ailleurs pas le résultat final pour notre exemple.

Remarquez que je n'ai pas posé le problème d'une incertitude initiale sur x et les coefficients de P ce qui constitue un problème d'erreur de méthode.

5. CONFIANCE A ACCORDER AUX CALCULS DES FONCTIONS DE L'ORDINATEUR

Pour l'instant, nous n'avons traité que de calculs arithmétiques effectués par l'ordinateur, nous savons qu'alors les résultats sont exacts autant qu'ils peuvent l'être : tronqués (ou arrondis) à 10 chiffres bien entendu.

Pour ce qui est des autres calculs, aussi longtemps que les constructeurs de microordinateurs se refuseront à fournir une table d'erreurs, nous ne pouvons que leur faire confiance et *assimilerons ces erreurs à des erreurs de troncature*. Les résultats semblent effectivement exacts sur les intervalles usuels, mais calculez donc : $\sin(\pi 10^{10})$, vous ne trouverez surement pas 0 ni un nombre qui en soit très proche.

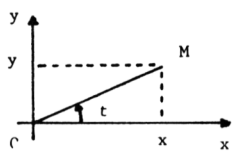
Le problème de ces calculs sera détaillé dans la deuxième partie du tome 2, pour l'instant nous allons voir un algorithme souvent utilisé pour le calcul des fonctions trigonométriques, ce qui élucidera également notre problème.

En exercices 8 et 9, vous trouverez également des algorithmes de calcul de l'exponentielle et du logarithme.

A. PRINCIPE DE L'ALGORITHME DE BRIGGS

Cet algorithme est du à Brigg (Mathématicien anglais, 1561-1630, qui l'a utilisé pour constituer des tables de valeurs numériques - à la main bien entendu !), de nos jours il est souvent appelé algorithme CORDIC.

Grâce aux formules de trigonométrie, on se ramène à calculer tg t pour $t \in [0, \frac{\pi}{4}]$



Si un point M du plan vérifie : $(\overrightarrow{Ox}, \overrightarrow{OM}) = t$ alors ses coordonnées x et y sont telles que :

$$\frac{y}{x} = \text{tg } t .$$

L'idée est alors de déterminer une suite de points M_n tels que : $(\overrightarrow{Ox}, \overrightarrow{OM_n}) = T_n$ avec : $\lim_{n \to +\infty} T_n = t$ et que les coordonnées de M_{n+1} soient simples à calculer en fonction de celles de M_n :

$$M_{n+1} = f_n(M_n)$$

où f_n est une similitude de centre O et d'angle $T_{n+1} - T_n$ d'où les formules :

$$\begin{cases} x_{n+1} = \lambda_n \cos(T_{n+1}-T_n)x_n - \lambda_n \sin(T_{n+1}-T_n)y_n \\ y_{n+1} = \lambda_n \sin(T_{n+1}-T_n)x_n + \lambda_n \cos(T_{n+1}-T_n)y_n \end{cases}$$

on choisit $\lambda_n = \frac{1}{\cos(T_{n+1}-T_n)}$ ce qui simplifie les formules :

$$\begin{cases} x_{n+1} = x_n - \text{tg}(T_{n+1}-T_n)y_n \\ y_{n+1} = \text{tg}(T_{n+1}-T_n)x_n + y_n \end{cases}$$

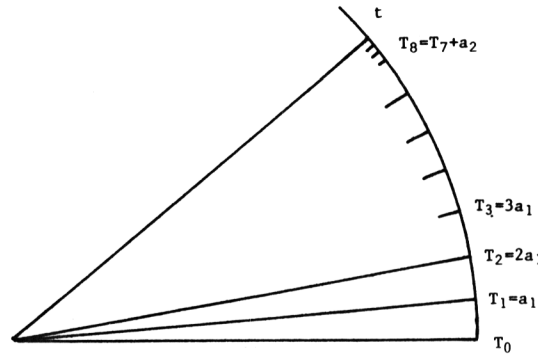
On évite toute multiplication en choisissant :

$$\text{tg}(T_{n+1}-T_n) = 10^{-j} \text{ où } j \text{ est un entier,}$$

ce qui permet de définir une construction de cette suite (T_n) :

B. ALGORITHME DE BRIGGS

On pose : $a_n = \text{Arctg}(10^{-n})$ pour $n \in \mathbb{N}^*$ c'est-à-dire que a_n est l'unique nombre de $[0, \frac{\pi}{2}]$ tel que : $\text{tg } a_n = 10^{-n}$. On enlève de t autant de fois a_1 que l'on peut, puis a_2 ... etc ... ce qui permet d'approcher t (remarquez l'analogie avec la division) :



On définit ainsi une suite (t_n) :

$$\begin{cases} t_0 = t \\ t_{n+1} = t_n - a_j \text{ si } a_j \text{ est le plus grand } a_j \text{ que l'on} \end{cases}$$

peut enlever de t_n, c'est-à-dire : $a_j < t_n \leqslant a_{j-1}$.

La suite (T_n) définie par : $T_n = t - t_n$ est donc une suite d'approximations de t.

Les formules :

$$\begin{cases} x_0 = 1 \quad y_0 = 0 \\ x_{n+1} = x_n - 10^{-j}\, y_n \\ y_{n+1} = 10^{-j}\, x_n + y_n \end{cases}$$

définissent les coordonnées d'une suite de points M_n tels que : $(\overrightarrow{Ox}, \overrightarrow{OM_n}) = T_n$.

Vous remarquerez que ces formules sont très bien adaptées à un ordinateur calculant dans le système décimal car elles ne comportent que des additions (ou des soustractions) et des décalages de virgule ce qui est beaucoup plus rapide qu'une multiplication.

D'autre part : $\lim_{n\to+\infty} T_n = t$ donc, par continuité, $\lim_{n\to+\infty} \frac{y_n}{x_n} = \text{tg } t$ ce qui justifie l'algorithme d'un point de vue qualitatif.

C. ANALYSE DE L'ALGORITHME

Cet algorithme exige de mettre en mémoire morte (MEM, ou ROM en anglais) la suite des a_j. En fait, il suffit de stocker a_1, a_2, a_3 et a_4 car dès a_5, $a_j = 10^{-j}$ avec une incertitude de 10^{-3j} [les premières valeurs sont

calculées avec le développement en série entière par exemple : voir la deuxième partie du tome 2] .

a. *Erreur de méthode*

Le problème est de savoir à quelle valeur de j arrêter cet algorithme. Supposons pour cela que cela soit pour N et que tous les calculs sont effectués de façon exacte.

Si n est le dernier indice, alors :

$$t - a_N \leqslant T_n \leqslant t$$

la fonction $t \longrightarrow \operatorname{tg} t$ étant croissante sur $[0, \frac{\pi}{4}]$,

$$\operatorname{tg}(t-a_N) \leqslant \frac{y_n}{x_n} \leqslant \operatorname{tg} t$$

$$\frac{\operatorname{tg} t - 10^{-N}}{1 + 10^{-N} \operatorname{tg} t} \leqslant \frac{y_n}{x_n} \leqslant \operatorname{tg} t$$

d'où l'incertitude due à la méthode :

$$\Delta_N = \operatorname{tg} t - \frac{\operatorname{tg} t - 10^{-N}}{1 + 10^{-N} \operatorname{tg} t}$$

$$= 10^{-N} \frac{1 + \operatorname{tg}^2 t}{1 + 10^{-N} \operatorname{tg} t}$$

or : $\operatorname{tg} t \in [0,1]$, donc l'incertitude est de 2.10^{-N}. Donc si on désire une incertitude relative d'au plus 10^{-10} pour les valeurs de t de $[10^{-3}, \frac{\pi}{4}]$, on prend N = 14.

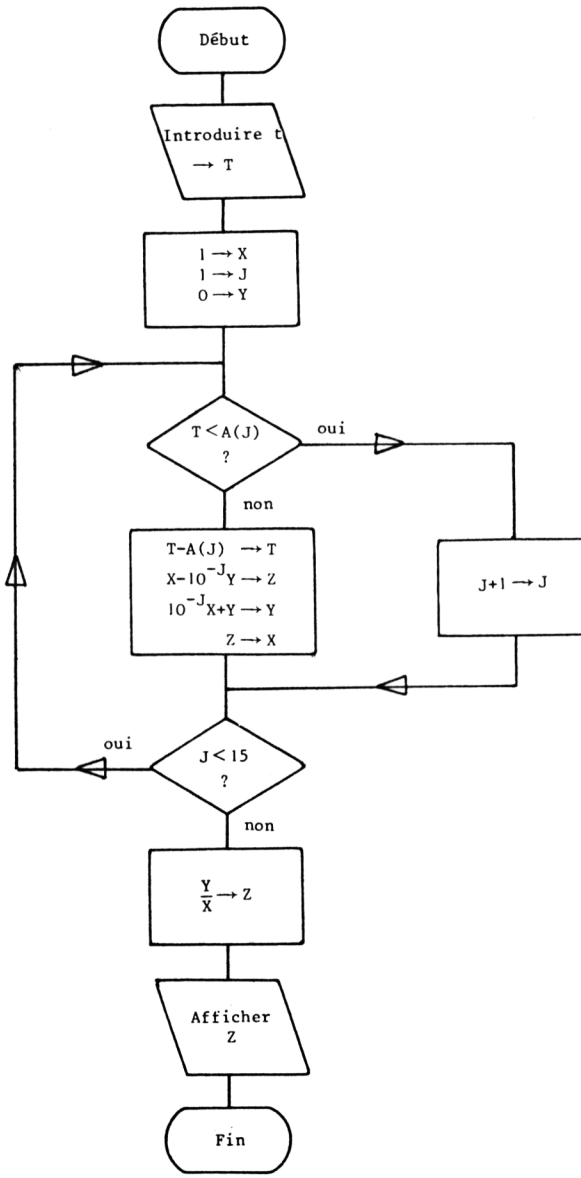
b. *Organigramme*

Pour discuter des erreurs de calcul, il est d'abord nécessaire d'écrire l'organigramme exact des calculs : (voir page suivante)

Remarquez que cet organigramme est analogue à celui de la division d'un nombre t par un nombre a.

La suite (a_j) est alors définie par : $a_j = a \times 10^{-j}$, j est initialisé à 0 et les suites (x_n^j) et (y_n^j) sont remplacées par la seule suite (y_n) définie par :

$$\begin{cases} y_0 = 0 \\ y_{n+1} = y_n + 10^{-j} \end{cases}$$


Début
Introduire t
→ T
1 → X
1 → J
0 → Y
T < A(J)
?
oui
non
T-A(J) → T
X-10^-J Y → Z
10^-J X+Y → Y
Z → X
J+1 → J
J < 15
?
oui
non
Y/X → Z
Afficher
Z
Fin

c. *Erreur de calcul*

Les calculs successifs des t_n sont approchés si bien que les calculs effectués n'ont pas le sens que leur attribue la théorie : la suite des itérations n'est pas celle prévue.

Afin d'analyser correctement les erreurs commises, il faut légèrement modifier l'approche théorique de l'algorithme : supposer la suite des t_n exacte, c'est-à-dire poser :

$$t_{n+1} = t_n - b_n$$

b_n est donc une valeur approchée de a_j, l'erreur étant due aux troncatures sur a_j et le calcul de $t_n - a_j$ donc :

$$|b_n - a_j| \leqslant 10^{-10}(|t_{n+1}| + |a_j|)$$

$|t_{n+1}|$ est majoré par a_{j-1}, lui-même de l'ordre de $10^{-(j-1)}$ donc :

$$|b_n - a_j| \leqslant 1{,}1.10^{-9-j}.$$

Les formules exactes :

$$\begin{cases} x_{n+1} = x_n - (\text{tg } b_n)y_n \\ y_{n+1} = (\text{tg } b_n)x_n + y_n \end{cases}$$

sont remplacées pour les calculs par :

$$\begin{cases} x_{n+1} = x_n - (\text{tg } a_j)y_n \\ y_{n+1} = (\text{tg } a_j)x_n + y_n. \end{cases}$$

D'après l'inégalité des accroissements finis, l'erreur sur tg b_n est majorée par :

$$|\text{tg } a_j - \text{tg } b_n| \leqslant |a_j - b_n| \sup_{x \in [0, a_0]} (1 + \text{tg}^2 x)$$

donc :

$$|\text{tg } a_j - \text{tg } b_n| \leqslant 1{,}2 \times 10^{-9-j}.$$

Donc si Δ_n est l'incertitude sur x_n et Δ'_n l'incertitude sur y_n :

$$\begin{cases} \Delta_{n+1} = \Delta_n + |y_n|\ 1{,}2 \times 10^{-9-j} + 10^{-j}\ \Delta'_n + |x_{n+1}|\,10^{-10} \\ \Delta'_{n+1} = 10^{-j}\ \Delta_n + |x_n|\ 1{,}2 \times 10^{-9-j} + \Delta'_n + |y_{n+1}|\,10^{-10} \end{cases}$$

les derniers termes $(|x_{n+1}|\ 10^{-10}$ et $|y_{n+1}|\ 10^{-10})$ représentant l'erreur de troncature.

Ceci exige d'évaluer x_n et y_n ; or : $OM_{n+1} = \frac{1}{\cos a_j} OM_n$ c'est-à-dire $OM_{n+1} = \sqrt{1 + 10^{-2j}}\ OM_n$. Il y a au plus $\frac{a_{j-1}}{a_j}$ étapes comportant l'angle a_j, c'est-à-dire 10, donc :

$$1 \leqslant OM_n \leqslant (1 + 10^{-2})^5 \ \ldots\ (1 + 10^{-2j})^5 \qquad \text{car } OM_0 = 1.$$

donc :
$$1 \leqslant OM_n \leqslant (1 + 10^{-2})^5 \ldots (1 + 10^{-28})^5$$
$$\leqslant 1 + 5 \sum_{j=1}^{14} 10^{-2j}$$
$$\leqslant 1{,}06$$

donc : $x_n \leqslant 1{,}06 \cos T \qquad y_n \leqslant 1{,}06 \sin T.$

Finalement, en majorant les erreurs :

$$\begin{cases} \Delta_{n+1} = \Delta_n + 1{,}1 \times 10^{-10} \\ \Delta'_{n+1} = \Delta'_n + 1{,}1 \times 10^{-10}. \end{cases}$$

Il y a au plus 140 itérations, donc : $\Delta x = \Delta y = 1{,}5 \times 10^{-8}$.

$$\Delta\left(\frac{y}{x}\right) = \frac{\Delta y}{x} + y\frac{\Delta x}{x^2}$$
$$= 2{,}2 \times 10^{-8} \text{ en majorant } y \text{ et } \frac{1}{x} \ \left(t \in [0, \tfrac{\pi}{4}]\right)$$

L'erreur de méthode est négligeable devant cette erreur donc l'erreur absolue sur le calcul de tg t pour $t \in [10^{-3}, \frac{\pi}{4}]$ est de $2{,}2 \times 10^{-8}$. Pour les petites valeurs de t, la formule : tg t = t est préférable car son erreur est majorée par t^3, on peut également utiliser : $\text{tg } t = t + \frac{t^3}{3}$ avec une erreur majorée par t^5.

En fait, mon estimation de l'erreur est très pessimiste : elle suppose que l'erreur de troncature est toujours maximale. En pratique, elle est en moyenne de 0,5 fois la dernière unité et non de 1. De plus le nombre d'itérations n'est pas de 140 en moyenne (ce qui est le cas le plus défavorable) mais de 70 ce qui donne une incertitude vraisemblable de 5×10^{-9}.

D. CONCLUSION

L'exemple précédent montre le type d'algorithmes utilisés sur les petits ordinateurs (et les calculatrices scientifiques). Ce n'est évidemment pas le seul possible, ni le meilleur du point de vue de la précision (voir la deuxième partie du tome 2), mais celui-ci a l'avantage d'être rapide d'exécution pour une machine calculant en décimal (codé binaire). Par contre, il est utopique de croire que l'erreur est systématiquement une erreur de troncature. Cela ne sera le cas, avec ce type d'algorithmes et pour des valeurs usuelles de l'argument, que si l'ordinateur calcule avec deux ou trois chiffres de plus qu'il n'en affiche.

6. EXERCICES

1. Soit $(u_n)_{n\geqslant 0}$ la suite définie par : $\begin{cases} u_0 = 2 \\ u_{n+1} = \sqrt[4]{u_n} \end{cases}$ pour tout $n \geqslant 0$

   Faites un programme qui calcule les valeurs successives de $\dfrac{u_{n+1}-1}{u_n-1}$.
   Que remarquez-vous ? Expliquez. Corrigez votre programme.

2. Calculez la valeur de : $\dfrac{1}{x+2} - \dfrac{3}{x} + \dfrac{2}{x+1}$ pour de grandes valeurs de x.

Les exercices suivants proposent des calculs de π :

3. Formule de Viete.
   Montrez que : $\lim\limits_{n\to+\infty} 2^n \sin \dfrac{\pi}{2^n} = \pi$.

   Soit $(u_n)_{n\geqslant 0}$ et $(v_n)_{n\geqslant 0}$ les suites définies par :

   $\begin{cases} u_0 = 0 \\ u_{n+1} = \sqrt{2+u_n} \end{cases}$ pour tout $n \geqslant 0$ $\qquad$ $\begin{cases} v_0 = 2 \\ v_{n+1} = 2v_n \end{cases}$ pour tout $n \geqslant 0$

   Montrez que : $\lim\limits_{n\to+\infty} v_n \sqrt{2-u_n} = \pi$.
   Etudiez l'erreur de méthode commise quand on remplace π par $v_n\sqrt{2-u_n}$.
   Programmez cette méthode de calcul de π. Que constatez-vous ? Expliquez.

4. Méthode des isopérimètres.
   Soit n un entier et P_n un polygône régulier à 2^n côtés de périmètre 2.
   Soit r_n et R_n les rayons des cercles inscrits et circonscrits à P_n.
   Montrez que : $\dfrac{1}{R_n} \leqslant \pi \leqslant \dfrac{1}{r_n}$ puis que :

$$r_{n+1} = \frac{r_n + R_n}{2} \text{ et } R_{n+1} = \sqrt{R_n\, r_{n+1}} \text{ pour tout } n \geqslant 2.$$

   Calculez π avec la meilleure précision possible. [Vous étudierez soigneusement les incertitudes de méthode et de calcul].

5. Formule de Wallis.
   Soit $(u_n)_{n\geqslant 0}$ la suite définie par : $u_n = \displaystyle\int_0^{\pi/2} \sin^n x\, dx$.

   Montrez la formule de récurrence : $n\,u_n = (n-1)u_{n-2}$ pour tout $n \geqslant 2$. Déduisez-en que :

$$u_{2n} = \frac{1.3.5 \ldots (2n-1)}{2.4.6 \ldots (2n)} \frac{\pi}{2}, \quad u_{2n+1} = \frac{2.4.6 \ldots (2n)}{1.3.5 \ldots (2n+1)}.$$

   Montrez que la suite $(u_n)_{n\geqslant 0}$ est décroissante puis déduisez-en que :

$$\lim_{n\to+\infty} \frac{\big(2.4.6 \ldots (2n)\big)^2}{\big(1.3.5 \ldots (2n-1)\big)^2 (2n+1)} = \frac{\pi}{2}$$

   Etudiez la méthode de calcul de Π qui s'en déduit, programmez-la.
   Qu'en pensez-vous ?

Le calcul de π sera repris dans la deuxième partie du tome 2 : nous y aboutirons au calcul d'un grand nombre de décimales de π.

6. Soit $(u_n)_{n\geqslant 0}$ la suite définie par :

$$\begin{cases} u_0 = 9 \qquad u_1 = \sqrt{101} \\ u_{n+2} = 11u_{n+1} - 5u_n - 55 \text{ pour tout } n \geqslant 0 . \end{cases}$$

Programmez le calcul de u_{100} puis étudiez la convergence de $(u_n)_{n\geqslant 0}$. Expliquez les résultats obtenus.

7. Programmez l'algorithme de Briggs pour le calcul de tg t $\left(t \in [0, \frac{\pi}{4}]\right)$. Comparez les erreurs réelles des calculs (pour des valeurs "connues" : tg $\frac{\pi}{3}$, tg $\frac{\pi}{6}$, tg $\frac{\pi}{12}$,etc.) et les incertitudes calculées.

8. *Calcul de l'exponentielle par la méthode de Briggs.*
Remarquez qu'il est simple de se ramener au calcul de e^t pour : $0 \leqslant t \leqslant \text{Log } 10$.
Soit $(a_n)_{n\in\mathbb{N}}$ la suite définie par : $a_n = \text{Log}(1 + 10^{-n})$, $(t_n)_{n\in\mathbb{N}}$ la suite définie par :

$$\begin{cases} t_0 = t \\ t_{n+1} = t_n - a_j \text{ si } a_j \text{ est le plus grand } a_j \text{ que l'on} \end{cases}$$

peut enlever de t_n, et $(x_n)_{n\in\mathbb{N}}$ la suite définie par :

$$\begin{cases} x_0 = 1 \\ x_{n+1} = x_n(1 + 10^{-j}) . \end{cases}$$

Montrez que la suite (x_n) converge vers e^t. Remarquez que cet algorithme a les mêmes avantages que celui exposé pour le calcul de tg t. Etudiez sa réalisation pratique et l'incertitude sur le résultat. Faites le programme et comparez vos résultats avec les résultats connus, vérifiez ainsi vos calculs d'incertitude.

9. *Calcul du logarithme par la méthode de Briggs*
Remarquez qu'il est simple de se ramener au calcul de Log t pour $t \in [1, 10]$.
Soit $(a_n)_{n\in\mathbb{N}}$ la suite définie par : $a_n = 1 + 10^{-n}$, $(t_n)_{n\in\mathbb{N}}$ la suite définie par :

$$\begin{cases} t_0 = t \\ t_{n+1} = t_n\, a_j \text{ si } a_j \text{ est le plus grand } a_j \text{ tel que } t_{n+1} \leqslant 10 \end{cases}$$

et $(x_n)_{n\in\mathbb{N}}$ la suite définie par :

$$\begin{cases} x_0 = \text{Log } 10 \\ x_{n+1} = x_n - \text{Log } a_j . \end{cases}$$

Montrez que la suite (x_n) converge vers Log t . Remarquez que cet algorithme a les mêmes avantages que celui exposé pour le calcul de tg t . Etudiez sa réalisation pratique et l'incertitude sur le résultat. Faites le programme et comparez vos résultats avec les résultats connus, vérifiez vos calculs d'incertitude.

10. Essayez d'adapter la méthode de Briggs aux fonctions transcendantes usuelles que nous n'avons pas étudiées : Arcsinus, Arcosinus, Arctangente.

11. Montrez que : $(1-t)(1+t+t^2+ \dots +t^n) = 1 - t^{n+1}$. Déduisez-en que si $|t| < 1$ alors $\lim_{n\to+\infty} (1+t+ \dots +t^n) = \frac{1}{1-t}$ ce qui s'écrit encore : $\sum_{n=0}^{\infty} t^n = \frac{1}{1-t}$.

Ceci a-t-il un intérêt pour le calcul de certains inverses ?

Les exercices suivants proposent un calcul de n!

12. Par définition, n! est le produit de $1 \times 2 \times 3 \times \dots \times n$ pour n entier. Faites un programme de calcul de n! (par itération). Jusqu'à quelle valeur de n, ce programme vous fournit-il un résultat ?

13. Formule de Stirling

Montrez que : $\int_1^n \text{Log } n\,dx \leqslant \text{Log } n! \leqslant \int_1^{n+1} \text{Log } n\,dx$ (Vous interpréterez ceci géométriquement). Déduisez-en un équivalent de Log n! . Soit : $u_n = n \text{ Log } n - n$ et $v_n = \text{Log } n! - u_n$, vérifiez que : $v_n = \sum_{p=1}^{n} w_p$ où : $w_p = \text{Log } p - (u_p - u_{p-1})$. Déterminez un développement limité de w_p et déduisez-en un développement asymptotique de u_n.

Montrez qu'il existe une constante $C > 0$ telle que : $n! \underset{n\to+\infty}{\sim} C\left(\frac{n}{e}\right)^n \sqrt{n}$. Calculez C à l'aide des formules de Wallis (voir l'exercice 5). Enfin, montrez que :

$$n! = \sqrt{2\pi n}\ \left(\frac{n}{e}\right)^n \left[1 + \frac{1}{12n} + \frac{1}{288n^2} + O\left(\frac{1}{n^2}\right)\right].$$

Déterminez l'incertitude sur n! en le remplaçant par :

$$\sqrt{2\pi n}\ \left(\frac{n}{e}\right)^n \left[1 + \frac{1}{12n} + \frac{1}{288n^2}\right].$$

14. Calcul de n! pour n grand.

Déduisez de la formule de Stirling (voir la fin de l'exercice 13) un calcul approché de n! Faites le programme.

Quel est le nombre de chiffres de 1000! ?

Faites un programme de calcul de n! valable pour toutes les valeurs de n. (et donnant 3! = 6 et non 5,8 ...).

15. Peut-on améliorer la formule de Stirling ? Est-ce intéressant pour le calcul de n! ? Améliorez vos programmes.

16. Examinez le problème des incertitudes dans les programmes de division de polynômes du chapitre 3.

QUATRIEME PARTIE

LA RÉSOLUTION DES ÉQUATIONS

La résolution des équations constitue sans doute le problème essentiel des mathématiques. Contrairement à une opinion très répandue, celle-ci ne passe pas en général par une formule explicite donnant les racines mais plutôt par une méthode approchée.

Le plus souvent, pour résoudre une équation, nous avons besoin d'une information sur l'ordre de grandeur des racines :

il est nécessaire de savoir que la racine cherchée se trouve dans un certain intervalle]a,b[.

Dans les applications, ceci n'est pas une limitation car nous avons en général une connaissance a priori de l'ordre de grandeur du résultat (que ce soit un calcul de pH ou de taux d'intérêt ou tout autre calcul d'origine concrète).

Dans le chapitre 7, nous voyons la méthode de dichotomie qui, pour être la plus élémentaire, n'en est pas moins souvent suffisante. Nous l'appliquons à quelques calculs (pH, taux d'intérêt).

Dans le chapitre 8, nous voyons une méthode plus performante : la méthode des approximations successives et ses améliorations (en particulier de Newton), nous l'appliquons à des calculs mathématiques (calcul de 20 décimales de $\sqrt{2}$, $\sqrt{3}$, etc ...) puis au calcul des positions planétaires.

Dans le chapitre 9, nous faisons un cas à part aux polynômes (racines réelles et imaginaires, simples et multiples) avec la méthode de Bairstow.

CHAPITRE 7

LA MÉTHODE DE DICHOTOMIE

Si une fonction f est continue et strictement monotone sur un intervalle [a,b] et si $f(a)f(b)<0$ alors l'équation : $f(x)=0$ a une racine unique sur [a,b].

Si bien que, si ε est un nombre positif, si $x_0 \in [a,b]$ et si $f(x_0-\varepsilon)f(x_0+\varepsilon)<0$ alors x_0 est la racine de l'équation avec une incertitude de ε.

Cette remarque est essentielle pour la résolution des équations numériques. Remarquez également dès à présent que le calcul de $f(x_0-\varepsilon)f(x_0+\varepsilon)$ étant, en général, approché, il est nécessaire que sa précision soit suffisante pour assurer que : $f(x_0-\varepsilon)f(x_0+\varepsilon)<0$.

C'est cette remarque qui porte la limitation essentielle de la méthode.

1. LA METHODE DE DICHOTOMIE

A. DESCRIPTION DE LA METHODE

La méthode consiste à considérer le milieu de [a,b] : $c=\frac{a+b}{2}$ et de tester le signe de f(a)f(c).

Si $f(a)f(c)<0$ alors la racine appartient à [a,c] sinon elle appartient à [c,b]. On construit donc la suite d'intervalles :

$$[a_0,b_0] = [a,b]$$

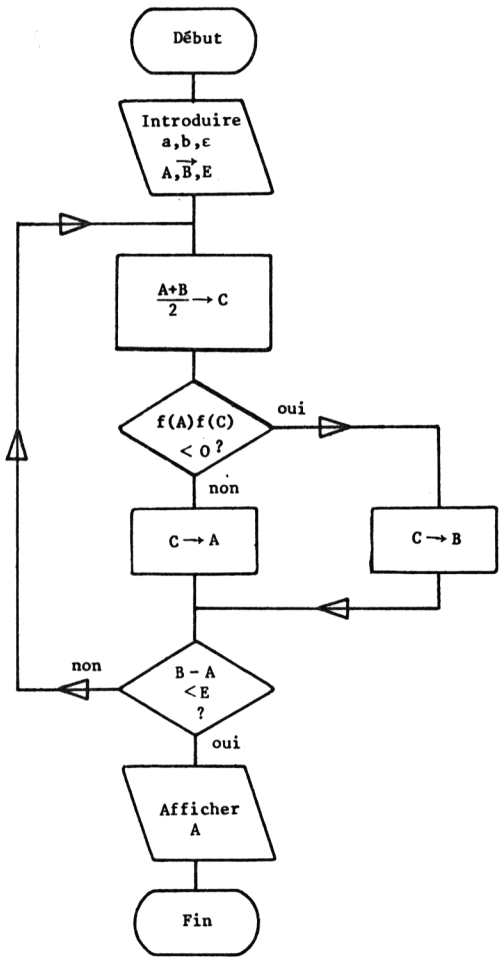
$$[a_1,b_1] = \begin{cases} [a,c] & \text{si } f(a)f(c)<0 \\ [c,b] & \text{sinon} \end{cases}$$

et ainsi de suite $[a_n,b_n]$.

La longueur de l'intervalle $[a_n,b_n]$ est $\frac{b-a}{2^n}$ donc si on désire une précision de ε, il suffit de construire $[a_n,b_n]$ pour n tel que :

$$\frac{b-a}{2^n} \leqslant \varepsilon.$$

B. ORGANIGRAMME



Le calcul de f(A) et f(C) se fait en sous-programme bien entendu ; pour le programme suivant, j'ai amélioré l'organigramme afin de ne pas calculer deux fois f(A).

C. LISTE DU PROGRAMME

```
 10 ! Resolution d'une equation
 20 !      par dichotomie
 30 !     ****************
 40 !
 50 ! Introduction des donnees
 60 ! ************************
 70 !
 80 INPUT A,B,E
 90 ! a,b=bornes de l'intervalle
100 ! E=precision
110 !
120 !     Initialisation:
130 !     ***************
140 ! calcul de f(a) et f(b)
150 !
160 C=A @ GOSUB 390 @ X=Z
170 C=B @ GOSUB 390 @ Y=Z
180 !
190 !     Boucle de calcul
200 !     ****************
210 !
220 C=(A+B)/2
230 GOSUB 390 ! calcul de f(c)
240 D=X*Z ! d=f(a).f(c)
250 IF D=0 THEN 330
260 IF D<0 THEN 280
270 A=C @ X=Z @ GOTO 290
280 B=C @ Y=Z
290 ! La precision est-elle
300 ! atteinte?
310 IF B-A>E THEN 190
320 !
330 !    Sortie du resultat
340 !    ******************
350 !
360 PRINT C
370 END
380 !
390 ! Sous-programme de calcul
400 !          de f(c)
410 Z=SIN(C)
420 RETURN
```

Les variables X,Y,Z désignent f(A), f(B) et f(C). Le sous-programme (ligne 390) calcule f(C) car c'est le calcul le plus courant.

Remarquez que le calcul est plus rapide sans sous-programme mais il faut alors écrire trois fois le calcul de f(x) ; remarquez également que, pour fixer les idées, la fonction f choisie est la fonction sinus.

Enfin, si votre ordinateur possède l'instruction : IF … THEN … ELSE, les lignes 260, 270 et 280 peuvent être remplacées par :

```
260 IF D<0 THEN B=C : Y=Z ELSE A=C : X=Z
```

D. EXECUTION DU PROGRAMME. PRECAUTION D'EMPLOI

a. *Exemples*

- Avec le programme tel qu'il est écrit, il est possible de calculer π :

Commandes	Affichage
RUN	?
3,4,.00001	3.141586303

Bien entendu, le résultat à retenir est 3,14159 à 10^{-5} près. On remarque qu'il est bien exact ce qui s'explique car à la dernière étape :

$$A = 3.141586303 \qquad X = 6.35 \times 10^{-6}$$
$$B = 3.14159\,3933 \qquad Y = -1.28 \times 10^{-6}$$

les erreurs sur X et Y sont manifestement inférieures à 10^{-7} d'où le résultat.

Par contre, si vous demandez une précision de 10^{-10}, le résultat est moins assuré.

- Calculons par exemple le zéro entre 1 et 3 de :

$$x^7 - 14x^6 + 84x^5 - 280x^4 + 560x^3 - 672x^2 + 448x - 128 = 0$$

dont les valeurs sont calculées à l'aide de l'algorithme de Hörner, il suffit de complèter le programme précédent par :

```
410 Z= -14+C
420 Z=  84+C*Z
430 Z=-280+C*Z
440 Z= 560+C*Z
450 Z=-672+C*Z
460 Z= 448+C*Z
470 Z=-128+C*Z
480 RETURN
```

l'exécution donne :

Commandes	Affichage
RUN	?
1,3,.0001	2

ce qui est bien le résultat car en fait l'équation s'écrit :

$$(x-2)^7 = 0.$$

Par contre

Commandes	Affichage
RUN	?
1.5,2.2,.0001	2.011840820

la précision véritable n'est que de 0,01 ; cela s'explique car à la dernière étape :

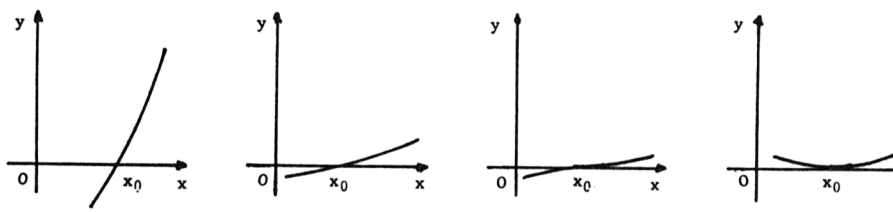
$$A = 2.011669921 \qquad X = -10^{-9}$$
$$B = 2.012011718 \qquad Y = 3 \times 10^{-9}$$

l'incertitude sur X est trop grande pour assurer $X<0$, d'où l'erreur.

b. *Conclusion*

Pour conclure une étude de ce type de façon rigoureuse, il est nécessaire, si x_0 est donné (à l'affichage de l'ordinateur) comme racine de : $f(x) = 0$ avec une incertitude de ε, de s'assurer que $f(x_0-\varepsilon)$ et $f(x_0+\varepsilon)$ sont de signes opposés, c'est-à-dire que l'incertitude sur les calculs de $f(x_0-\varepsilon)$ et $f(x_0+\varepsilon)$ permet de l'affirmer.

L'origine de ce problème se voit bien si on se réfère au graphe de la fonction f :



Si la pente de la courbe est importante au niveau de la racine x_0, le changement de signe de f au passage de x_0 est facile à déceler : la méthode permet alors une bonne précision (premier cas de figure).

Dans les cas de figures 2 et 3, ce passage est plus difficile à déceler aussi la précision est faible. Enfin dans le quatrième cas, la méthode n'est plus applicable car f ne change pas de signe : ce cas sera repris au chapitre 2 du tome 3 (à propos de l'optimisation : ici x_0 réalise également un extremum de f) et également au chapitre 8 de ce tome.

Tout ceci va être encore précisé sur les exemples suivants.

2. RESOLUTIONS D'EQUATIONS

A. UN EXEMPLE EN MATHEMATIQUES

Soit à déterminer les racines réelles de l'équation : $x^3 - x + 1 = 0$ avec une incertitude de 10^{-5}.

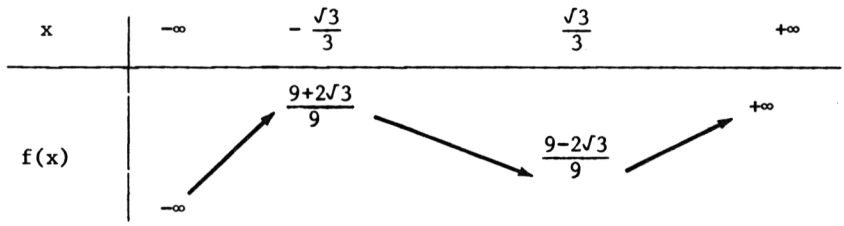
Il s'agit d'abord de faire une étude qualitative permettant de préciser le nombre et de localiser les racines de cette équation. Pour cela, je considère la fonction réelle $f : \mathbb{R} \to \mathbb{R}$ définie par :

$$f(x) = x^3 - x + 1.$$

Cette fonction est dérivable sur $\mathbb{R}$ et, pour tout x :

$$f'(x) = 3x^2 - 1 = 3\left(x - \frac{\sqrt{3}}{3}\right)\left(x + \frac{\sqrt{3}}{3}\right)$$

d'où les variations de f :



$\frac{9-2\sqrt{3}}{9} > 0$ donc : $f(x) = 0$ a une racine unique et celle-ci appartient à $]-\infty, -\frac{\sqrt{3}}{3}[$.

Je remarque que : $f(-1) = 1 > 0$ et $f(-2) = -5 < 0$ donc la racine appartient à $]-2,-1[$.

Je complète le programme du paragraphe 1.C. par :

```
410 Z=-1+C*C
420 Z= 1+C*Z
430 RETURN
```

Puis j'exécute le programme pour : -2, -1,.00001, j'obtiens : - 1.324714660. La valeur soupçonnée est : $x_0 = -1,32471$, je dois tester les signes de f(-1,32472) et f(-1,32470).

$$f(-1,32472) = -\,8,7116 \times 10^{-6} \qquad f(-1,32470) = 7,65798 \times 10^{-5}$$

Examinons les erreurs :

A la ligne 410, l'erreur sur Z est de : $|C^2-1| \times 10^{-10}$ c'est-à-dire : $\Delta Z = 0,75 \times 10^{-10}$ puis à la ligne 420 :

$$\Delta = |C|\Delta Z + |f(c)| \times 10^{-10}$$
$$= 10^{-10}$$

Donc, il est possible d'assurer que :

$$f(-1,32472) < 0 \qquad \text{et} \qquad f(-1,32470) > 0$$

donc la racine est : $x_0 = -1,32471$ avec une incertitude de 10^{-5}.

B. CALCUL D'UN TAUX D'INTERET

Vous pouvez rencontrer des annonces du genre :

"Emportez votre voiture pour 785 F par mois".

Et vous pouvez vous demander : quel est le taux d'intérêt de ce prêt ?

Le paiement dure quatre ans et la voiture en question vaut 24 870 F.

Je note t le taux d'intérêt mensuel et u_n le montant du prêt restant à rembourser après n mois. On a :

$$u_0 = 24\ 870 \qquad u_{48} = 0,$$

d'autre part : $u_{n+1} = u_n(1+t) - 785$ pour tout n ; car après un mois, il reste à rembourser le capital u_n augmenté des intérêts moins le montant du remboursement.

Successivement, je trouve :

$$u_0 = 24\ 870$$
$$u_1 = 24\ 870\ (1+t) - 785$$
$$u_2 = 24\ 870\ (1+t)^2 - 785\,[(1+t) + 1]$$

il semble donc que :

$$u_n = 24\ 870\ (1+t)^n - 785\,[(1+t)^{n-1} + (1+t)^{n-2} + \ldots + 1] \text{ pour tout n,}$$

c'est-à-dire que :

$$u_n = 24\ 870\ (1+t)^n - 785\ \frac{(1+t)^n - 1}{t}$$

(voir l'exercice 10 du chapitre 6), formule que l'on vérifie aisément par récurrence.

En particulier, pour n = 48 :

$$24\ 870\ (1+t)^{48} - 785\ \frac{(1+t)^{48} - 1}{t} = 0.$$

Equation qu'il est facile de résoudre avec notre programme, il suffit de le complèter par :

```
410 T = (1+C) ^ 48
420 Z = 24870*T-785*(T-1)/C
430  RETURN
```

Puis j'exécute le programme pour : .0001,.1,.00001 (car un taux d'intérêt mensuel ne saurait être hors de cet intervalle). La valeur est : $1,84 \times 10^{-2}$. Le taux d'intérêt annuel est donc de : $(1 + 1,84 \times 10^{-2})^{12} - 1$ c'est-à-dire 24,50% approximativement.

Remarquez que certains pourraient prétendre que celui-ci est de : $12 \times 1,84 \times 10^{-2}$ c'est-à-dire de 22,10% mais ceci ne correspond pas au taux réel mais à un taux fictif que les banquiers appellent le taux proportionnel.

C. CALCULS CHIMIQUES

Soit à calculer le pourcentage d'ionisation d'une solution à une mole par litre d'acide cyanhydrique (sa constante de dissociation est $K_a = 4,8 \times 10^{-10}$).

L'équilibre s'écrit :

$$HCN \rightleftarrows H^+ + CN^- \qquad K_a = \frac{[H^+]\ [CN^-]}{[HCN]}$$

où $[H^+]$ désigne la concentration en ions H^+ en moles par litre, de même pour $[CN^-]$ et $[HCN]$.

Les ions H^+ et CN^- étant présents en solution, leurs concentrations sont égales. Soit $x = [H^+] = [CN^-]$, donc : $[HCN] = 1 - x$, d'où :

$$\frac{x^2}{1-x} = 4,8 \times 10^{-10}$$

d'où : $x^2 + 4,8 \times 10^{-10}x - 4,8 \times 10^{-10} = 0$.

Equation qu'il est facile de résoudre avec notre programme, il suffit de le complèter par :

```
 05 T=4.8E-10
410 Z=T+C
420 Z=-T+C*Z
430 RETURN
```

puis j'exécute le programme pour : 0,1,.000001 d'où : $x = 2,19 \times 10^{-5}$ moles par litre, le pourcentage ionisé est donc approximativement 0,002%.

Examinons à ce propos une faute de raisonnement commise dans certains ouvrages de chimie :

Arrivé à l'équation : $\frac{x^2}{1-x} = 4,8 \times 10^{-10}$, l'auteur écrit que x est négligeable devant 1 donc : $x^2 = 4,8 \times 10^{-10}$ d'où le résultat. Ce raisonnement n'est pas faux en lui-même : il suffit de le vérifier en portant dans l'équation de départ : $2,18 \times 10^{-5}$ et $2,20 \times 10^{-5}$.

Mais bien souvent, l'auteur se contente d'écrire que, en effet, le résultat $x = 2,19 \times 10^{-5}$ est négligeable devant 1 et que ceci justifie l'hypothèse.

Voici un raisonnement du même type. Soit l'équation :

$$\frac{1+x^2}{1-x} = 1 + 10^{-10}$$

Supposons x négligeable devant 1 alors l'équation se simplifie en :

$$x^2 = 10^{-10} \quad \text{d'où} \quad x = \pm\ 10^{-5}.$$

L'hypothèse est bien vérifiée !

Je vous laisse voir le vice de raisonnement, la fausseté du résultat ainsi que l'approximation correcte à faire.

3. EXERCICES

1. Résolvez les équations :

$$x^3 - 5x^2 + 3x + 2 = 0$$
$$x^5 + 4x^3 + 2x - 1 = 0$$
$$\cos x - x = 0$$

Dans chacun des cas, déterminez avec soin la meilleure précision que vous pouvez obtenir.

Etudiez le signe des fonctions $x \longrightarrow x^3 - 5x^2 + 3x + 2$, $x \longrightarrow x^5 + 4x^3 + 2x - 1$, $x \longrightarrow \cos x - x$.

2. Reprenez les questions de l'exercice 1, pour l'équation :

$$343x^3 - 441x^2 + 189x - 27 = 0.$$

Trouvez ses racines exactes en cherchant ses racines rationnelles (voir le chapitre 4).

3. Quel est le nombre de calculs à effectuer pour déterminer la racine de : $f(x) = 0$ sur $[a,b]$ avec une incertitude ε par la méthode de dichotomie ?

Par un découpage en trois ? en quatre ? etc … .

Quel est le procédé le plus rapide ? Comparez également avec la méthode de balayage (essais successifs de a, $a + \varepsilon$, $a + 2\varepsilon$, … etc …).

4. Résolvez l'équation : $x^3 - 3x - 4 = 0$.

Il est également possible de résoudre cette équation de façon exacte. Pour cela, on pose : $u + v = x$; $uv = 1$.

Montrez que x est solution de l'équation si et seulement si : $u^3 + v^3 = 4$, déduisez-en u^3 et v^3 puis x.

Retrouvez la valeur de x. Qu'en concluez-vous ? Cette formule exacte a-t-elle un avantage par rapport à la méthode approchée ?

Déterminez les racines complexes de l'équation.

5. Montrez que toute équation du troisième degré se met sous la forme : $x^3 + px + q = 0$ (p et q constantes réelles) au moyen d'une translation sur x (c'est-à-dire un changement de variable du type : $X = x + h$ où h est une constante).

On pose : $u + v = x$; $uv = \alpha$. Déterminez α pour que l'équation soit équivalente à une équation : $u^3 + v^3 = \beta$ où β est une constante que vous déterminerez.

Déduisez en u^3 et v^3 puis les racines (vous discuterez suivant les valeurs de p et q le nombre de racines réelles).

Déterminez des formules donnant les racines de l'équation en fonction de p et q. Faites un programme les utilisant. Comparez son utilisation avec l'utilisation directe du programme de dichotomie.

6. M. Gaspard obtient un prêt de 200 000 F sur une période de dix ans pour l'achat d'un appartement. Il est remboursable par trimestrialités au taux proportionnel de 17,50% (voir le paragraphe 2.B.). La loi lui permet de déduire 7 000 F d'intérêts de son revenu imposable pendant dix ans. Sachant que l'économie d'impôts annuelle est de 2 100 F (tranche à 30%), calculez l'intérêt réel de l'emprunt.

7. Le capitaine Haddock s'achète un nouveau château au centre de Paris à 10 millions de francs, pour cela il doit revendre Moulinsart.

Il a l'opportunité de le vendre immédiatement à 3 millions de francs, mais il voudrait le vendre à 5 millions de francs.

Il pense qu'à ce prix, il a une chance sur deux de le vendre en un an, trois sur quatre de le vendre en deux ans et il est sûr de le vendre en trois ans. Pour financer l'achat de son nouveau château, il doit prendre un crédit-relai sur la vente de Moulinsart à 18% d'intérêt annuel.

Que feriez-vous à sa place ?

Remarquez que ce type de problème se pose à beaucoup de propriétaires.

8. Deux échelles AC et BD reposent sur un sol horizontal AB et s'appuient sur des murs verticaux AD et BC. Elles se croisent en un point I situé à un mètre au-dessus du sol. De plus : DB = 3m et AC = 2m. Déterminez la longueur AB.

9. Une chèvre est attachée à un piquet A situé sur la circonférence d'un champ circulaire de rayon 20m. Calculez la longueur à donner à la corde pour que la chèvre puisse brouter la moitié du champ.

10. *Recherche dichotomique*

Un tableau A$(.) de chaînes toutes distinctes et triées dans l'ordre lexicographique (ordre des dictionnaires) étant donné, chercher une chaine B$ dans le tableau A$(.) consiste à chercher l'indice I tel que : B$ = A$(I).

Adaptez l'idée de la résolution des équations par dichotomie à ce problème.

Cette question a une grande importance pour les logiciels de consultation de banques de données.

11. Quels sont les invariants des boucles des programmes de ce chapitre ?

CHAPITRE 8

LA MÉTHODE DES APPROXIMATIONS SUCCESSIVES ET AMÉLIORATIONS

La méthode de dichotomie est simple mais parfois longue. Dans ce chapitre, nous allons voir des algorithmes en général plus rapides et de plus très stables ; ils sont tous dérivés de la *méthode des approximations successives*.

Cette méthode provient de la remarque suivante :

Si φ est une fonction continue de $\mathbb{R}$ dans $\mathbb{R}$, a un réel et si la suite $(u_n)_{n\geqslant 0}$ définie par :

$$\begin{cases} u_0 = a \\ \quad \text{et} \\ u_{n+1} = \varphi(u_n) \text{ pour tout } n \geqslant 0 \end{cases}$$

converge vers un réel x, alors on a : $\varphi(x) = x$.

Cette remarque permet de résoudre des équations précédemment mises sous la forme : $\varphi(x) - x = 0$ (x est alors appelée un *point fixe* de φ).

Dans le paragraphe 1, nous voyons dans quelles conditions la suite $(u_n)_{n\geqslant 0}$ converge et étudions sa rapidité de convergence - ce problème est d'ailleurs lié au calcul de l'incertitude sur la racine. Puis au paragraphe 2, nous voyons comment transformer une équation donnée : $f(x) = 0$ en une équation sous la forme : $\varphi(x) = x$ et en particulier la méthode de Newton.

Notons dès maintenant que cette méthode est également utilisée pour résoudre des systèmes d'équations (voir le chapitre 11), des équations différentielles (ordinaires ou aux dérivées partielles) comme nous le verrons au tome 3 et qu'elle a également d'importantes applications théoriques en analyse (pour cela, vous pouvez consulter l'ouvrage de Jean Dieudonné : Fondements de l'Analyse Moderne, chapitre 10).

1. LA METHODE DES APPROXIMATIONS SUCCESSIVES

A. ETUDE D'UN EXEMPLE

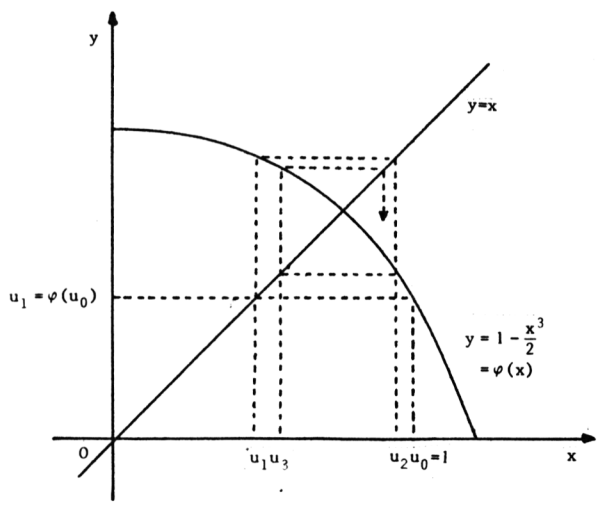
Soit à résoudre l'équation : $x^3 + 2x - 2 = 0$.

Une étude qualitative (variations de $x \longrightarrow x^3 + 2x - 2$) montre que cette équation n'a qu'une solution et que celle-ci appartient à $]0,1[$. Pour l'écrire sous la forme : $\varphi(x) = x$,

- une première idée est d'écrire : $x = 1 - \frac{x^3}{2}$. L'exécution du programme :

```
10 X=1
20 X=1-X*X*X/2
30 PRINT X
40 GOTO 20
```

semble indiquer une convergence lente vers 0,77 (après 50 itérations). Il suffit alors de calculer $x^3 + 2x - 2$ pour 0,76 et 0,78 pour assurer que la racine est 0,77 avec une incertitude de 10^{-2}. La convergence est cependant fort lente comme le montre le graphe :



Remarquez la construction géométrique des termes de la suite $(u_n)_{n \geqslant 0}$ définie par :

$$\begin{cases} u_0 = 1 \\ \text{et} \\ u_{n+1} = 1 - \frac{u_n^3}{2} \text{ pour tout } n \geqslant 0 \end{cases}$$

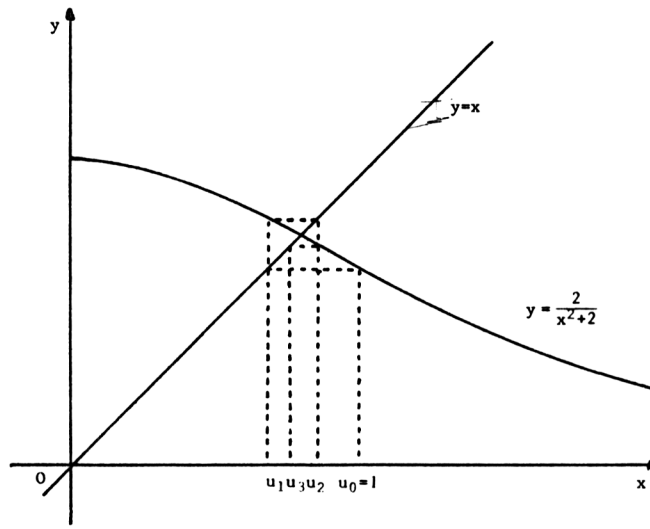
- Une autre idée est d'écrire l'équation sous la forme : $x = \frac{2}{x^2 + 2}$, l'exécution du programme précédent en remplaçant la ligne 20 par :

```
20 X=2/(X*X+2)
```

donne une convergence qui semble rapide :

0,77 est obtenu au bout de 8 itérations, dès 30 itérations on obtient : 0,7709169971 [ce qui se vérifie par les calculs de $x^3 + 2x - 2$ en 0,7709169970 et 0,7709169972 par l'algorithme de Hörm, sans oublier un calcul d'incertitude !] .

Le graphe suivant montre la rapidité de la convergence :



• Remarquez que le procédé peut également diverger comme le montre l'exemple de : $x^3 + 3x - 2 = x$.

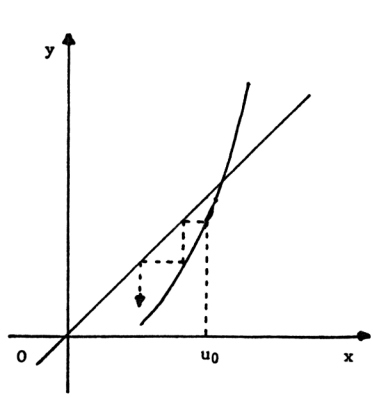
B. RAPIDITE DE LA CONVERGENCE

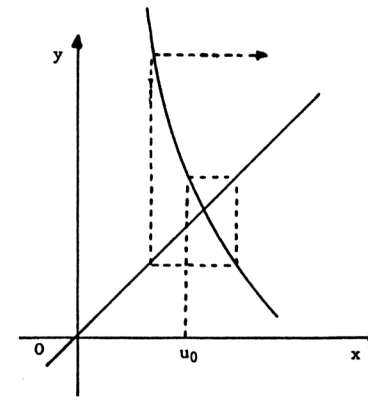
a. *Etude expérimentale*

Nous voyons donc expérimentalement que la rapidité de la convergence dépend essentiellement de la pente de la tangente au graphe de φ au voisinage du point fixe.

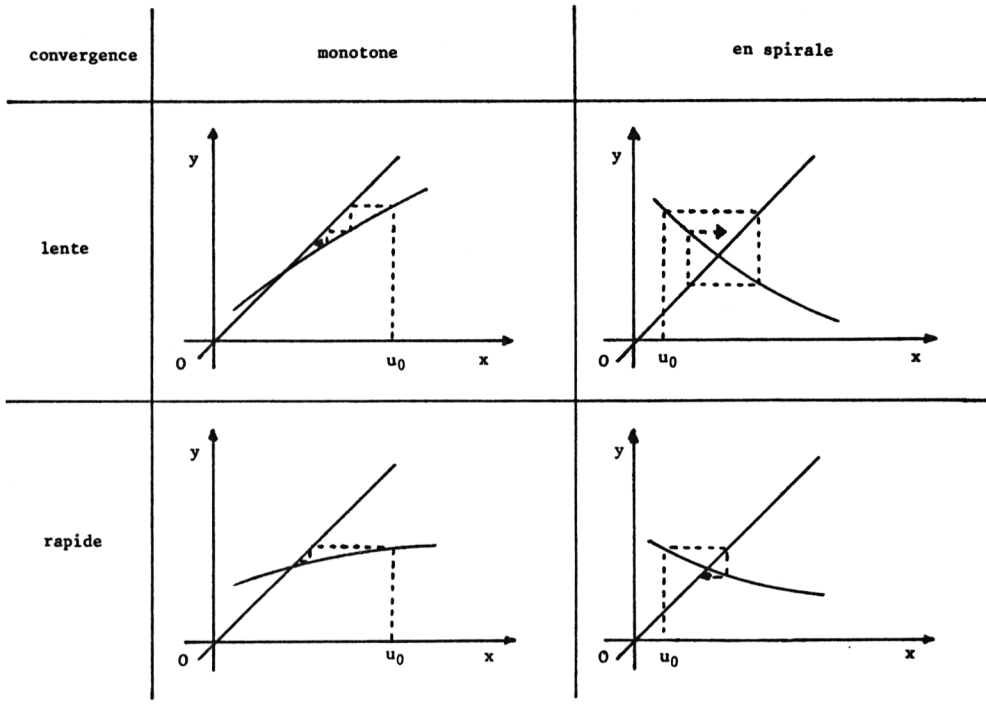
Voici les différents cas de figures :

Points fixes répulsifs :



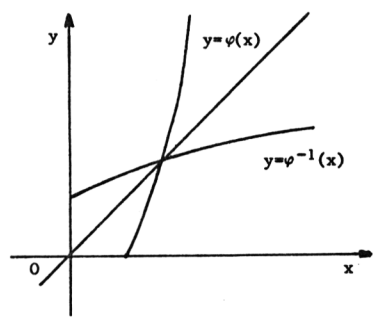


Points fixes attractifs



Il ressort de cette étude que l'idéal est que la tangente au point fixe soit horizontale.

Remarquez que si le point fixe de φ est répulsif, celui de φ^{-1} (si cette application est bien définie) est attractif et c'est le même. Nous utiliserons cette remarque pour calculer des racines de : tgx = x (voir l'exercice 4) :



b. *Etude théorique*

Le *théorème du point fixe* précise l'étude expérimentale précédente.

Soit φ : [a,b] $\longrightarrow$ [a,b] une fonction de classe C^1 sur [a,b] telle que :

$$\|\varphi'\|_\infty < 1$$

c'est-à-dire que : $\sup_{x\in[a,b]} |\varphi'(x)| < 1$.

Dans ce cas φ a un point fixe et un seul x_0. Celui-ci est obtenu comme limite de la suite $(u_n)_{n\geqslant 0}$ définie par :

$$\begin{cases} u_0 \in [a,b] & \text{choisi arbitrairement, et} \\ u_{n+1} = \varphi(u_n) & \text{pour tout } n \geqslant 0. \end{cases}$$

De plus, si : $\|\varphi'\|_\infty \leqslant k < 1$ alors, pour tout $n \geqslant 1$:

$$|u_n - x_0| \leqslant \frac{k^n}{1-k} |u_1 - u_0|$$

Nous retrouvons bien que, plus k est petit c'est-à-dire plus la pente de la tangente au voisinage du point fixe est faible, plus la convergence est rapide.

• Le premier point de la démonstration de ce théorème est que, d'après l'inégalité des accroissements finis

$$|\varphi(x) - \varphi(y)| \leqslant k|x-y| \text{ pour tout } (x,y) \in [a,b]^2.$$

• Puis je démontre que la série $\sum(u_{n+1}-u_n)$ est absolument convergente, car :

$$|u_{n+1} - u_n| \leqslant k|u_n - u_{n-1}| \text{ pour tout } n \geqslant 1$$

et donc, par récurrence :

$$|u_{n+1} - u_n| \leqslant k^n |u_1 - u_0| \text{ pour tout } n \geqslant 0.$$

Or la série $\sum k^n$ est convergente ($k \in [0,1[$, voir l'exercice 10 du chapitre 6), donc la série $\sum(u_{n+1}-u_n)$ est absolument convergente donc convergente.

La suite $(u_n)_{n\geqslant 0}$ est donc convergente, soit x_0 sa limite. φ est continue donc : $\lim_{n\to+\infty} \varphi(u_n) = \varphi(x_0)$ et donc, par unicité de la limite, $\varphi(x_0) = x_0$.

• L'unicité du point fixe est triviale, car si x en est un :
$|x_0 - x| \leqslant k|x_0 - x|$ et $k < 1$ implique : $|x_0 - x| = 0$ c'est-à-dire : $x = x_0$.

• Enfin, voici la majoration de $|u_n - x_0|$:

$$u_n - x_0 = \sum_{p=n}^{+\infty} (u_{p+1} - u_p) \text{ pour tout } n \geqslant 0, \text{ donc :}$$

$$|u_n - x_0| \leqslant \sum_{p=n}^{+\infty} |u_{p+1} - u_p|$$

$$\leqslant |u_1 - u_0| \sum_{p=n}^{\infty} k^p, \text{ donc :}$$

$$|u_n - x_0| \leqslant \frac{k^n}{1-k} |u_1 - u_0| \text{ pour tout } n \geqslant 0$$

Remarquez que cette démonstration reste valable si $\varphi : F \longrightarrow F$ où F est un fermé de $\mathbb{C}$, $\mathbb{R}^2$, $\mathbb{R}^3$ ou plus généralement $\mathbb{R}^p$, ou encore plus généralement d'un espace de Banach, l'hypothèse sur φ étant :

$$\|\varphi(x) - \varphi(y)\| \leqslant k \|x - y\| \text{ pour tout } (x,y) \in F^2$$

où $k \in [0, 1[$.

Nous l'appliquerons sous cette forme au chapitre 11 et dans le tome 3.

c. *Stabilité*

Heuristiquement, la stabilité de cet algorithme par rapport aux erreurs de calculs est due à ce que la convergence ne dépend pas du choix du premier terme u_0. Si bien qu'il n'est pas grave de remplacer u_n par une valeur approchée $\bar{u}_n$ à chaque itération.

Une étude plus précise peut cependant être faite car si Δ_n est l'incertitude sur u_n, alors, d'après l'inégalité des accroissements finis, l'incertitude de méthode sur u_{n+1} est $k\Delta_n$. Il convient de lui ajouter l'incertitude due aux calculs sur $\varphi(u_n)$ soit ε, la suite $(\Delta_n)_{n \geqslant 0}$ est alors définie par :

$$\begin{cases} \Delta_0 = 0 \\ \quad \text{et} \\ \Delta_{n+1} = k\Delta_n + \varepsilon \text{ pour tout } n \geqslant 0 \end{cases}$$

d'où, par récurrence :

$$\Delta_n = \varepsilon \frac{1 - k^n}{1 - k} \leqslant \frac{\varepsilon}{1 - k} \text{ pour tout } n \geqslant 0.$$

D'où finalement l'erreur totale sur x_0 :

$$|\bar{u}_n - x_0| \leqslant \frac{k^n |u_1 - u_0| + \varepsilon}{1 - k} \text{ pour tout } n \geqslant 0.$$

Nous constatons que les erreurs dues aux calculs ne se cumulent pas, l'algorithme est donc stable.

2. AMELIORATIONS

A. METHODE DE NEWTON

a. Une équation : $f(x) = 0$ étant donnée, l'idée est de se ramener à une équation : $\varphi(x) = x$ telle que φ' s'annule au point fixe x_0.

Pour cela, je cherche φ sous la forme :

$$\varphi(x) = x + \lambda(x)f(x).$$

En effet, si λ ne s'annule pas, $f(x) = 0$ si et seulement si $\varphi(x) = x$. Je suppose λ dérivable, alors φ l'est et :

$$\varphi'(x) = 1 + \lambda'(x)f(x) + \lambda(x)f'(x) \text{ pour tout } x$$

donc $\varphi'(x_0) = 0$ équivaut à $\lambda(x_0) = -\dfrac{1}{f'(x_0)}$.

Le choix de Newton pour obtenir cette condition est : $\lambda = -\dfrac{1}{f'}$, ce qui suppose que f' ne s'annule pas.

Notez que, pour ce qui nous intéresse, il est souvent plus pratique de choisir pour λ une constante proche de $-\dfrac{1}{f'(x_0)}$ (inconnue par hypothèse).

La suite ainsi construite est :

$$\begin{cases} u_0 \in I \\ \quad \text{et} \\ u_{n+1} = u_n - \dfrac{f(u_n)}{f'(u_n)} \text{ pour tout } n \geqslant 0 \end{cases}$$

ou bien :

$$\begin{cases} u_0 \in I \\ \quad \text{et} \\ u_{n+1} = u_n + af(u_n) \text{ pour tout } n \geqslant 0 \end{cases}$$

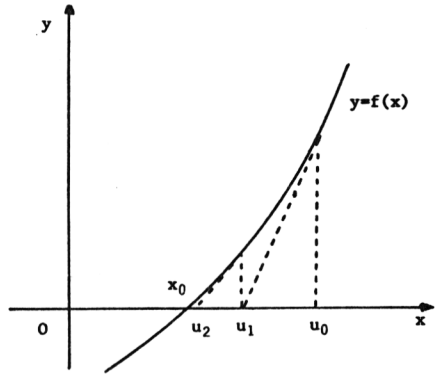
avec $a = -\dfrac{1}{f'(u_0)}$ par exemple.

Cette deuxième suite ne correspond pas à la méthode de Newton proprement dite mais à une de ses modifications.

Dans le même ordre d'idée, il est possible de chercher φ telle que : $\varphi'(x_0) = \varphi''(x_0) = 0$ ce qui, en principe, donne un algorithme plus rapide. Mais le calcul de φ est alors long et, en général, fait perdre son intérêt à la méthode (voir l'exercice 5).

b. *Interprétation géométrique*

• Vous remarquerez facilement que la méthode de Newton s'interprète de la façon suivante :



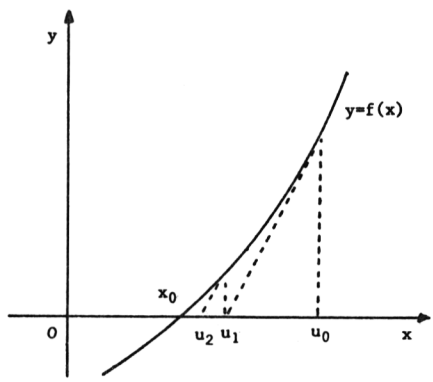
Car l'équation de la tangente au point d'abscisse u_n est :

$$y = f(u_n) + (x-u_n)f'(u_n)$$

donc, si $f'(u_n) \neq 0$, elle coupe l'axe Ox au point d'abscisse :

$$u_n - \frac{f(u_n)}{f'(u_n)} = u_{n+1} \ .$$

- De même, pour sa modification :



- La méthode de Newton correspond à une *linéarisation* de l'équation par le calcul différentiel ; en effet en première approximation :

$$f(x) = f(x_0) + (x-x_0)f'(x_0)$$

donc : $f(x) = 0$ équivaut à : $x = x_0 - \frac{f(x_0)}{f'(x_0)}$.

B. CONVERGENCE

La méthode de Newton est la méthode des approximations successives appliquée à la fonction φ définie par :

$$\varphi(x) = x - \frac{f(x)}{f'(x)} .$$

Nous nous ramènerons donc au théorème du point fixe pour son étude théorique ; une étude plus fine de la convergence est proposée en exercices 8, 9 et 10.

a. *Etude expérimentale*

• Reprenons l'exemple de l'équation : $x^3 + 2x - 2 = 0$. La méthode de Newton consiste à utiliser la fonction φ définie par :

$$\varphi(x) = x - \frac{x^3 + 2x - 2}{3x^2 + 2}$$
$$= 2\,\frac{x^3 + 1}{3x^2 + 2}$$

c'est-à-dire à transformer le programme du paragraphe 1.A en remplaçant la ligne 20 par :

```
20 X=2*(X*X*X+1)/(3*X*X+2)
```

le résultat avec 10 décimales exactes est obtenu après 4 itérations au lieu de 30 (voir paragraphe 1.A.).

La méthode de Newton modifiée consiste à utiliser la fonction φ définie par :

$$\varphi(x) = x - \frac{x^3 + 2x - 2}{5}$$
$$= \frac{-x^3 + 3x + 2}{5} \text{ , d'où :}$$

```
20 X=(2+X*(3-X*X))/5
```

le résultat est maintenant obtenu après 15 itérations mais chaque itération demande moins de calculs.

Donc si f' est délicat à calculer, la seconde méthode peut être préférable.

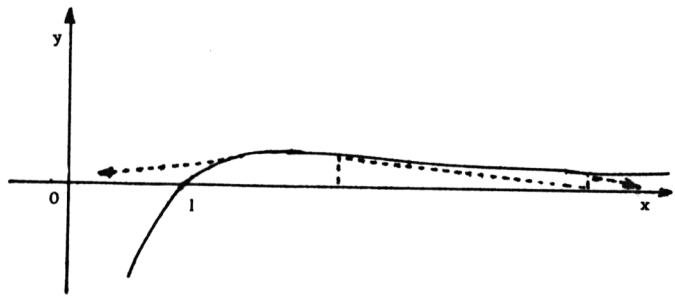
• La méthode peut donner des résultats inattendus si l'itération ne commence pas assez près du zéro :

Prenons l'exemple de l'équation : $\frac{x}{1+x^2} - \frac{1}{1+x} = 0$,

$$\varphi(x) = \frac{-3x^4 + 2x^3 + 2x^2 + 2x + 1}{2(-x^3+x^2+x+1)} .$$

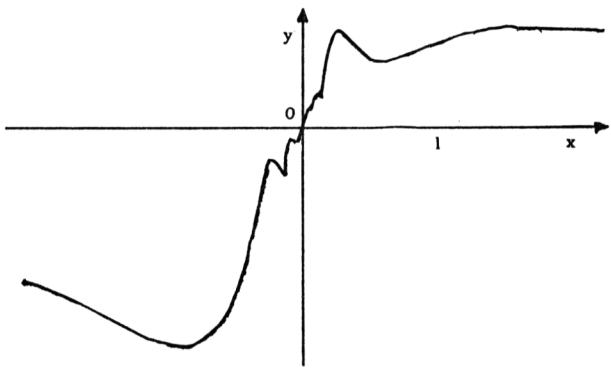
En commençant l'itération avec $x = 2$, il y a divergence vers $+\infty$, avec $x = 1{,}6$ divergence vers $-\infty$. A partir de 1,5 la suite converge effectivement vers la racine 1.

Ce phénomène s'explique si on examine le graphe de f :



• Les résultats sont bien plus curieux pour : $\frac{2x}{1+x^2} - \frac{x}{1+x^2} \sin \frac{1}{x} = 0$ qui a une racine unique 0.

Pour $x = 1$, il y a divergence vers $-\infty$, pour $x = 0{,}1$ divergence vers $+\infty$, à partir de $x = 0{,}01$ la suite semble constante. Tout ceci s'explique bien avec le graphe de f :

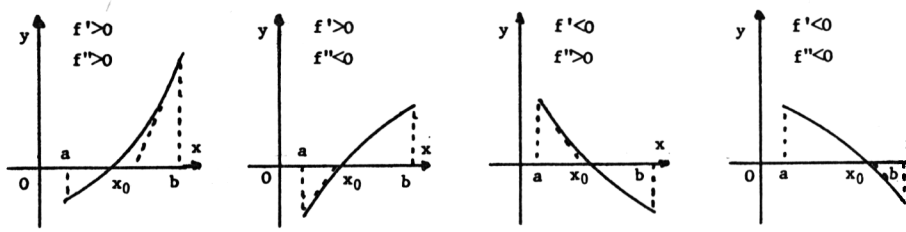


Les exercices 17 et 18 donnent d'autres exemples d'échecs de la méthode de Newton.

b. *Etude théorique*

Si f est une fonction de classe C^2 sur un intervalle [a,b] telle que $f(a)f(b) < 0$ et que f' et f" ne s'annulent pas sur [a,b] c'est-à-dire si l'on est dans l'un des quatre cas de figures ci-après.

Alors l'équation : $f(x) = 0$ a une racine unique x_0 sur [a,b] et la méthode des approximations successives pour la fonction φ définie par : $\varphi(x) = x - \frac{f(x)}{f'(x)}$ converge si on prend pour premier terme de la suite a si $f'f'' < 0$ et b si $f'f'' > 0$ (voir les figures).



Ce résultat est assez évident graphiquement, l'exercice 6 en propose une démonstration rigoureuse.

Dans les cas où nous aurons besoin d'une majoration de l'erreur (voir par exemple le paragraphe 3), nous utiliserons le théorème du point fixe.

c. *Conclusion*

Pour résoudre une équation : $f(x) = 0$, il est possible d'utiliser la méthode de Newton (ou de Newton-modifiée) sans se soucier, a priori, de sa précision. Vous arrêtez les itérations quand les décimales que vous désirez se sont stabilisées (c'est-à-dire restent les mêmes à chaque itération).

Si vous obtenez une valeur x_0 avec une précision espérée de ε, il vous suffit de calculer $f(x_0-\varepsilon)$ et $f(x_0+\varepsilon)$ pour vérifier ce résultat.

Si votre ordinateur "ne sort pas de la boucle", dites-vous que vous êtes dans un cas analogue à ceux qui ont été décrits au paragraphe 2.B.a :

l'itération n'a pas commencé assez près de la racine ou alors vous venez de rencontrer un spécimen de la tératologie mathématique.

Peut-être également avez-vous négligé l'étude qualitative du nombre de racines et de leur localisation ?

C. QUELQUES PROGRAMMES

Conformément aux remarques qui ont été faites, les programmes qui suivent commencent tous par une dichotomie.

Les entrées sont donc :

- l'intervalle [a,b], A et B
- la précision sur la méthode de dichotomie F
- la précision sur la méthode de Newton ou Newton-modifiée E (ε).

Le test d'arrêt est : $|u_n-u_{n-1}| < \varepsilon$ ce qui est justifié si φ vérifie le théorème du point fixe avec $k \leqslant 0,5$ mais n'assure par réellement la précision demandée aussi les sorties sont :

la racine (supposée) c et les valeurs de $f(c-\varepsilon)$ et $f(c+\varepsilon)$ ce qui, associé à un calcul d'incertitude, permet de vérifier le résultat.

a. *Méthode de Newton*

```
 10 ! Resolution d'une equation
 20 ! ******** (1) ************
 30 !         *******
 40 ! Introduction des donnees
 50 ! ************************
 60 !
 70 INPUT A,B,F,E
 80 ! a,b=bornes de l'intervalle
 90 ! F=precision sur la
100 ! methode de dichotomie.
110 ! E=precision sur la
120 ! methode de Newton.
130 !
140 !        Dichotomie
150 !        **********
160 C=A @ GOSUB 440 @ X=Z
170 C=B @ GOSUB 440 @ Y=Z
180 C=(A+B)/2
190 GOSUB 440
200 D=X*Z
210 IF D<0 THEN 230
220 A=C @ X=Z @ GOTO 240
230 B=C @ Y=Z
240 IF B-A>F THEN 180
250 !
260 !    Methode de Newton
270 !    *****************
280 GOSUB 440
290 ! Calcul de f'(c)
300 T=5*EXP(5*C)-1
310 D=Z/T
320 IF ABS(D)<E THEN 380
330 C=C-D
340 !
350 GOTO 280
360 !    Sortie du resultat
370 !    ******************
380 PRINT C
390 !       Verification
400 C=C-E @ GOSUB 440 @ PRINT Z
410 C=C+2*E @ GOSUB 440 @ PRINT
    Z
420 END
430 !
440 !      Calcul de f(c)
450 Z=EXP(5*C)-C-100
460 RETURN
```

Pour l'écriture du programme, j'ai pris l'exemple de la fonction f définie par : $f(x) = e^{5x} - x - 100$.

Pour une autre fonction, il suffit de changer les lignes 450 et 300.

b. *Méthode de Newton-modifiée*

Il existe énormément de modifications possibles de la méthode de Newton, chacune ayant un avantage particulier. Si f'(x) est long ou difficile à calculer, on peut le calculer une seule fois (à l'aide d'une formule exacte ou d'une formule approchée : voir le tome 2), d'où les deux programmes :

```
 10 ! Resolution d'une equation
 20 ! ********* (2) ***********
 30 !          *******
280 !     Calcul de f'(c)
290 T=5*EXP(5*C)-1
300 !       Iteration
310 GOSUB 440
320 D=Z/T
330 IF ABS(D)<E THEN 380
340 C=C-D
350 GOTO 300
```

```
 10 ! Resolution d'une equation
 20 ! ********* (3) ***********
 30 !          *******
280 !      Calcul de f'(c)
281 H=.00001
282 !    Calcul de f(c-h)
283 C=C-H @ GOSUB 440 @ X=Z
284 !    Calcul de f(c+h)
285 C=C+2*H @ GOSUB 440 @ Y=Z
286 !       Calcul de
287 !    [f(c+h)-f(c-h)]/2h
288 T=(Y-X)/(2*H)
289 !    Restauration de c
290 C=C-H
```

Cette dernière version ayant l'avantage d'éviter le calcul "manuel" de f'.

La liste du programme (2) ne comporte que la partie modifiée par rapport au programme (1), de même la liste du programme (3) ne comporte que la partie modifiée par rapport au programme (2).

c. *Etude comparative des temps d'exécution*

Sur l'exemple de l'équation : $e^{5x} - x - 100 = 0$ sur $[0,1]$ à 10^{-10} près, la dichotomie demande 2,5s à mon ordinateur et les autres méthodes (avec dichotomie à 0,1) 1s.

Pour les équations les plus usuelles, le gain de temps est de 2 à 4 mais ce n'est pas général : essayez les exemples du paragraphe 2.B.a. De même, vous pouvez refaire les calculs du chapitre 7 avec ces nouveaux programmes.

3. EXEMPLE DE CALCUL EN GRANDE PRECISION

A. CALCUL DE $\sqrt{2}$.

Votre ordinateur "connaît" 10 chiffres significatifs de $\sqrt{2}$, comment en obtenir 18 ? (et plus).

La méthode de Newton conduit à l'algorithme :

$$\begin{cases} u_0 = 1{,}414\ 213\ 562 \\ \text{et} \\ u_{n+1} = \dfrac{u_n}{2} + \dfrac{1}{u_n} \text{ pour tout } n \geqslant 0. \end{cases}$$

Soit φ la fonction définie par : $\varphi(x) = \frac{x}{2} + \frac{1}{x}$, φ applique $[u_0, \sqrt{2}]$ dans lui-même, φ' est croissante et négative sur $[u_0, \sqrt{2}]$ donc :

$$\|\varphi'\|_\infty = |\varphi'(u_0)| = \frac{2 - u_0^2}{2u_0^2}.$$

Or : $\sqrt{2} - 10^{-9} \leqslant u_0 \leqslant \sqrt{2}$ donc : $\|\varphi'\|_\infty \leqslant 0{,}8 \times 10^{-9}$.

Donc le théorème du point fixe est applicable et :

$$|u_n - \sqrt{2}| \leqslant \frac{(0{,}8 \times 10^{-9})^n}{1 - 0{,}8 \times 10^{-9}} |u_1 - u_0| \text{ pour tout } n \geqslant 1.$$

Le calcul de u_1 ou de u_2 doit suffire, il reste à calculer l'inverse d'un nombre en double précision.

Pour cela, j'utilise à nouveau la méthode de Newton pour le calcul de $\frac{1}{1{,}414\ 213\ 562}$:

$$\begin{cases} v_0 = 0,707\ 106\ 781 \\ \quad \text{et} \\ v_{n+1} = v_n(2 - 1,414\ 213\ 562\ v_n) \quad \text{pour tout } n \geqslant 0 \end{cases}$$

Soit ψ la fonction définie par : $\psi(x) = x(2-u_0x)$, ψ applique $[v_0, \frac{1}{u_0}]$ dans lui-même, ψ' est décroissante et positive sur $[v_0, \frac{1}{u_0}]$ donc :

$$\|\psi'\|_\infty = 2(1-u_0v_0) \leqslant 1,1 \times 10^{-9}.$$

Donc le théorème du point fixe est applicable et :

$$|v_n - \frac{1}{u_0}| \leqslant \frac{(1,1 \times 10^{-9})^n}{1 - 1,1 \times 10^{-9}} |v_1 - v_0| \text{ pour tout } n \geqslant 1.$$

Je calcule donc v_1 à l'aide du programme du chapitre 5 :

$$v_1 = 1,414\ 213\ 562 - 1,414\ 213\ 562 \times (0,707\ 106\ 781)^2$$
$$= 0,707\ 106\ 781\ 373\ 095\ 048\ 65 \text{ à } 10^{-20} \text{ près.}$$

donc :

$$|v_1 - v_0| \leqslant 4 \times 10^{-10} \text{ et donc : } |v_1 - \frac{1}{u_0}| \leqslant 5 \times 10^{-19}.$$

Donc :

$$u_1 = 0,707\ 106\ 781 + 0,707\ 106\ 781\ 373\ 095\ 0486\ 5$$
$$= 1,414\ 213\ 562\ 373\ 095\ 049 \text{ à } 10^{-18} \text{ près.}$$

donc :

$$|u_1 - u_0| \leqslant 4 \times 10^{-10} \text{ et donc } |u_1 - \sqrt{2}| \leqslant 4 \times 10^{-19}$$

donc finalement :

$\sqrt{2} = 1,414\ 213\ 562\ 373\ 095\ 049$ avec une incertitude de 10^{-18}.

Quelques itérations supplémentaires donnent un grand nombre de décimales de $\sqrt{2}$.

B. QUELQUES RESULTATS

Par ce procédé, il est facile d'obtenir 40 décimales de $\sqrt{2}$:

$\sqrt{2} = 1,41421\ 35623\ 73095\ 04880\ 16887\ 24209\ 69807\ 85697$

de même :

$\sqrt[3]{2} = 1,25992\ 10498\ 94873\ 16476\ 72106\ 07278\ 22835\ 05703$

$\sqrt[4]{2} = 1,18920\ 71150\ 02721\ 06671\ 74999\ 70560\ 47591\ 52930.$

Résultats qui vous permettront de vérifier vos calculs si vous réalisez un logiciel de calculs arithmétiques en grande précision comme le propose l'exercice 14.

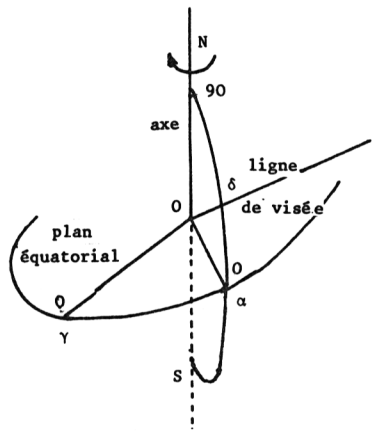
4. CALCULS ASTRONOMIQUES

Dans ce paragraphe, nous nous proposons de calculer l'éphéméride d'une planète de façon à permettre le réglage d'un télescope à monture équatoriale.

Pour des démonstrations complètes des formules données, vous pouvez vous reporter à l'ouvrage d'A. DANJON : Astronomie générale.

A. LE TELESCOPE

Le télescope est maintenu dans un repère indépendant de la rotation terrestre grâce à une monture équatoriale (c'est-à-dire sur un plan parallèle à l'équateur) et un moteur qui fait tourner cette monture de 1 tour par 24 heures dans le sens opposé à celui de la terre.

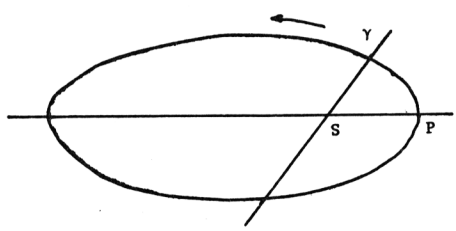


Les astres sont repérés par des coordonnées sphériques dont le plan fondamental est l'équateur et l'origine des longitudes l'axe Oγ qui est fixe par rapport au ciel et que l'on définira plus loin. La longitude α est appelée ici l'ascension droite et est mesurée en heures et la latitude δ la déclinaison mesurée en degrés : α varie de 0 à 24 et δ de -90 à +90.

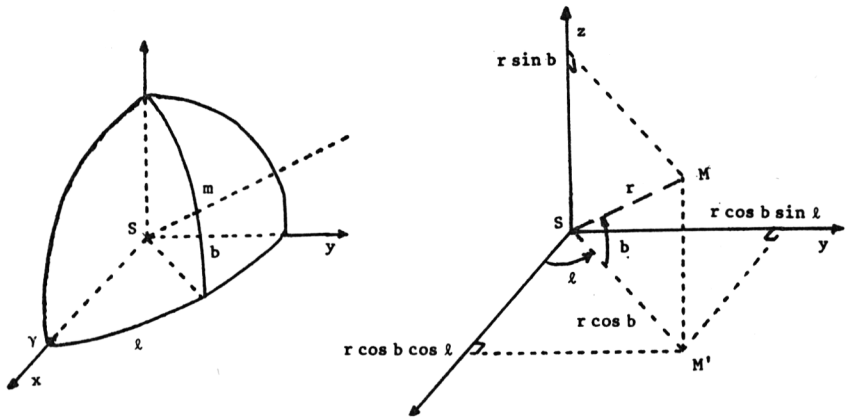
B. LE MOUVEMENT DES PLANETES

a. *Les coordonnées écliptiques héliocentriques*

La trajectoire de la terre est une ellipse de foyer le soleil S appelé l'écliptique ; son grand axe le coupe en deux points dont le plus proche de S est le périhélie P. Le plan de l'équateur a une direction constante par rapport à ce plan : ils font un angle $\varepsilon = 23{,}439°$ et le plan parallèle à l'équateur passant par S coupe l'écliptique en deux points : l'équinoxe de printemps ou point vernal γ et l'équinoxe d'automne.



Un astre est repéré par rapport au soleil par des coordonnées sphériques dont le plan fondamental est le plan de l'écliptique et l'origine des longitudes l'axe Sγ. Sur la figure, les points sont tous rapportés à une sphère de centre S :

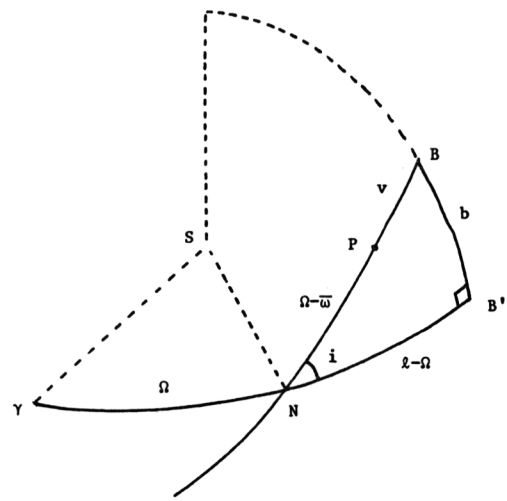


l'angle (ou l'arc) ℓ est la longitude de M, b sa latitude et r = SM son rayon vecteur (ℓ varie de 0° à 360° et b de -90° à +90°).On a les formules de passage :

(1) $x = r \cos b \cos \ell \qquad y = r \cos b \sin \ell \qquad z = r \sin b.$

b. *Le mouvement des planètes*

La trajectoire d'une planète B est une ellipse de foyer S, elle est essentiellement déterminée par les cinq paramètres : e,a,i,Ω et ϖ . e étant son excentricité, a son demi grand axe, i l'angle entre le plan de la trajectoire et le plan de l'écliptique, Ω la longitude de leur intersection et ϖ la longitude du périhélie de la planète. La figure représente ces points portés sur une sphère de centre S.



Si v est l'angle $\widehat{PB}$ (c'est-à-dire $(\overrightarrow{SP},\overrightarrow{SB})$), le triangle sphérique NB'B étant rectangle en B', en posant : $\omega = \varpi - \Omega$

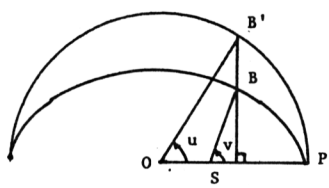
$$(2)\qquad \begin{cases} \sin b = \sin i \sin(v+\omega) \\ \cos b \cos(\ell-\Omega) = \cos(v+\omega) \\ \cos b \sin(\ell-\Omega) = \cos i \sin(v+\omega) \end{cases}$$

Ce qui fournit le sinus et le cosinus de b et ℓ en fonction de v, plus précisément, on obtient les coordonnées cartésiennes de B en éliminant b et ℓ entre les équations (1) et (2) :

$$(3)\qquad \begin{cases} x = (\cos\Omega\cos\omega - \cos i \sin\Omega \sin\omega) r\cos v + (-\cos\Omega\sin\omega - \cos i \sin\Omega\cos\omega) r \sin v \\ y = (\sin\Omega\cos\omega + \cos i \cos\Omega \sin\omega) r\cos v + (-\sin\Omega\sin\omega + \cos i \cos\Omega\cos\omega) r \sin v \\ z = (\sin i \sin\omega) r\cos v + (\sin i \cos\omega) r \sin v \end{cases}$$

Nous allons voir comment obtenir v en fonction du temps :

c. *L'équation de Kepler*



Plaçons-nous dans le plan de la trajectoire, soit O le centre de l'ellipse et C le cercle de centre O passant par P ; à B on fait correspondre le point B' comme indiqué sur la figure puis $u = (\overrightarrow{OP},\overrightarrow{OB'})$:

L'intérêt de cet angle u est d'être solution de l'équation de Kepler :

$$u - e \sin u = L - \varpi$$

où L est une "constante" liée à la planète.

On démontre alors facilement que :

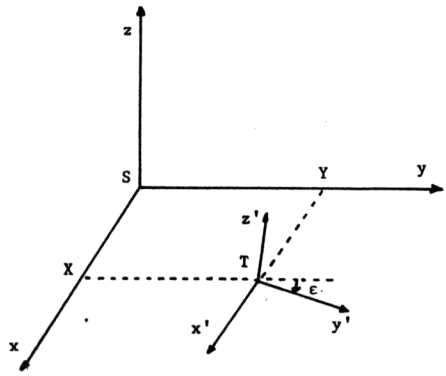
$$(4)\qquad r\cos v = a(\cos u - e) \qquad r \sin v = a\sqrt{1-e^2}\sin u.$$

d. *Calcul de l'ascension droite et de la déclinaison*

• Connaissant les coordonnées écliptiques héliocentriques (x,y,z) d'une planète et (X,Y,0) de la terre, on en déduit les coordonnées équatoriales de la planète par les formules de changement de repère :

$$(5)\qquad \begin{cases} x' = x - X \\ y' = (y-Y)\cos\varepsilon - z\sin\varepsilon \\ z' = (z-Y)\sin\varepsilon + z\cos\varepsilon \end{cases}$$

qui correspondent à la translation $\overrightarrow{ST}$ suivie de la rotation de $-\varepsilon$ autour de $\overrightarrow{Tx'}$:



• On en déduit alors α et δ :
α est l'angle polaire du point de coordonnées (x',y') et δ celui du point de coordonnées $\left(\sqrt{x'^2+y'^2},z'\right)$.

C. PROGRAMME

a. *Principe*

Les constantes L,ϖ,Ω,e,i et a n'ont de constantes que le nom, ce sont en fait des fonctions du temps t dont on connaît approximativement les premiers termes du développement en série entière. Vu leurs faibles variations (à notre échelle!), je me suis contenté des deux premiers termes pour L, ϖ et Ω et du premier seulement pour e, i et a. Ces constantes ont été calculées en radians à partir de tables fournies par le Bureau des Longitudes et ramenées à une origine des temps au 0 Janvier 1901 et à une unité d'un jour ce qui permet d'utiliser le programme du chapitre 1 pour le calcul de t. Ceci n'est pas les conventions usuelles mais simplifie le programme ; en particulier, le calcul en radians permet une résolution de l'équation de Kepler par la méthode des approximations successives ce que ne permet pas le calcul en degrés (voir l'exercice 13).

Le tableau des constantes planétaires est entré sous la forme d'un tableau de nombres A(I,J) où J désigne la planète et I le numéro de la constante. Il est entré en début de programme.

Les approximations choisies suffisent pour le réglage d'un télescope d'amateur.

b. *Liste du programme*

Je vous laisse le soin de déchiffrer ce programme. A la ligne 500, la fonction ATN2(X,Y) donne l'angle polaire du point de coordonnées (X,Y) : elle ne porte pas ce nom sur tous les ordinateurs, on rencontre souvent RPC(X,Y).

En entrée, le programme demande l'heure (TU), le jour, le mois, l'année puis le numéro de la planète :

```
 10 !  Calcul des ephemerides
 20 ! ************************
 30 !       Initialisation
 40 !      ***************
 50 !   Tableau des constantes
 60 !         planetaires
 70 FOR J=1 TO 8 @ FOR I=1 TO 9
 80 READ A(I,J)
 90 NEXT I @ NEXT J
100 DATA 4.01166,.071425454,1.32
    493,.000000742289,.823045,.0
    00000566185,.205615,.122225,
    .387099
110 DATA 3.60861,.027963119,2.27
    1616,.00000065572,1.32291,.0
    00000436681,.006816,.0592301
    ,.723332
120 DATA 1.72727,.0172028,1.7668
    8,.000000818559,0,0,.016751,
    0,1
130 DATA 2.17756,.0091467658,5.8
    3378,.000000879297,.851616,.
    000000371232,.093309,.032293
    9,1.5236
140 DATA 4.68279,.00145099,.2289
    ,.000000857,1.73578,.0000004
    82933,.048376,.0228418,5.202
    799
150 DATA 4.8567,.00058484,1.5974
    ,.000000412,1.96856,.0000004
    17308,.054311,.0435026,9.552
    098
160 DATA 4.3224,.000205424,2.952
    3,.000000762,1.2825,.0000002
    38237,.047319,.013482,19.216
    94
170 DATA 1.5223,.000105061,.7637
    ,.000000393,2.28102,.0000005
    2517,.008262,.0310536,30.129
    12
180 !  Tableau des constantes
190 !       calendaires
200 FOR I=1 TO 12
210 READ B(I)
220 NEXT I
230 DATA 0,31,59.25,90.25,120.25
    ,151.25,181.25,212.25,243.25
    ,273.25,304.25,334.25
240 !
250 ! Introduction des donnees
260 ! ************************
270 !
280 INPUT H,J,M,A@ INPUT N
290 !      Calcul de t
300 !     ************
310 T=365.25*(A-1901)+B(M)+J
320 T=INT(T)+H/24
330 !
340 !  Calcul des coordonnees
350 !        de la Terre
360 J=3 @ GOSUB 620
370 G=X @ H=Y ! Conservation des
     coordonnees
380 !  Calcul des coordonnees
390 !        de la Planete
400 !       ***************
410 J=N @ GOSUB 620
420 !   Calcul des coordonnees
430 ! equatoriales geocentriques
440 !   ********************
450 X=X-G @ Y=Y-H
460 T=Y*.917484-Z*.397772 @ Z=Y
    .397772+Z*.917484 @ Y=T
470 !      Ascension droite
480 !       et declinaison
490 !       ************
500 A=ATN2(Y,X) @ D=ATN2(Z,SQR(
    *X+Y*Y))
510 !       Conversion
520 A=A*12/PI
530 IF A<0 THEN A=24+A
540 H=INT(A) @ M=INT(60*(A-H))
550 D=D*180/PI
555 A$="N" @ IF D<0 THEN A$="S"
560 D=ABS(D) @ B=INT(D) @ C=INT
    60*(D-B))
570 !     Sortie du resultat
580 !   ********************
590 PRINT "Ascension droite";H;
600 PRINT "Declinaison";B;C;A$
610 END
620 ! Calcul des coordonnees
630 !      d'une planete
640 !        *********
650 !  Calcul des constantes
660 !        planetaires
670 L=A(1,J)+A(2,J)*T
680 O=A(3,J)+A(4,J)*T
690 P=A(5,J)+A(6,J)*T
700 E=A(7,J)
710 I=A(8,J)
720 A=A(9,J)
730 ! Resolution de l'equation
740 !        de Kepler
750 M=L-O @ U=M
760 FOR J=1 TO 10
770 U=M+E*SIN(U)
780 NEXT J
790 !       Formule (3)
800 R=A*(COS(U)-E)
810 V=A*SQR(1-E*E)*SIN(U)
820 O=O-P @ M=SIN(O) @ O=COS(O
830 Q=SIN(P) @ P=COS(P)
840 J=SIN(I) @ I=COS(I)
850 X=(P*O-I*Q*M)*R+(-P*M-I*Q*
    *V
860 Y=(Q*O+I*P*M)*R+(-Q*M+I*P*
    *V
870 Z=J*M*R+J*O*V
880 RETURN
```

Mercure (1), Vénus (2), Mars (4), Jupiter (5), Saturne (6), Uranus (7), Neptune (8).

En sortie, il donne l'ascension droite en heures et minutes et la déclinaison en degrés et secondes (Nord ou Sud).

c. *Utilisation*

Soit à déterminer les positions des planètes (connues des anciens) le 10 Mars 1982 à 6 H TU, l'exécution du programme donne :

	Ascension droite	Déclinaison
Mercure	21 H 51 mn	14°45 mn S
Vénus	20 H 26 mn	14°57 mn S
Mars	13 H 8 mn	3°45 mn S
Jupiter	14 H 32 mn	13°30 mn S
Saturne	13 H 22 mn	5°42 mn S

Cette conjonction est le signe de la fin du Monde pour certains astrologues. Pouvez-vous prédire le jour de la prochaine fin du Monde ?

Au tome 4, nous utiliserons ce programme pour tracer le mouvement apparent des planètes.

5. EXERCICES

1. Résolvez les équations :

$$\text{sh } x - 100x = 0$$
$$x - 30\text{Log } x - 100 = 0$$
$$x^3 + x - 1 = 0$$
$$x - \text{Log } x - 2 = 0$$

Dans chacun des cas, déterminez avec soin la meilleure précision que vous pouvez obtenir. Comparez les temps d'exécution suivant la méthode utilisée.

2. L'équation : $e^{5x} = x + 100$ s'écrit également : $x = \frac{1}{5}\text{Log}(x+100)$. Comparez.

3. Créez un logiciel de résolution des équations de la forme : $e^{ax} + bx + c = 0$. L'entrée sera a,b,c et la sortie les racines.

4. Montrez que les racines positives de l'équation : $\text{tg } x = x$ forment une suite $(u_n)_{n\geq 0}$ strictement croissante.

Calculez u_0, u_1, u_2 avec la meilleure précision possible. Comparez les diverses méthodes.

Déterminez une suite $(an+b)_{n\geq 0}$ telle que : $\lim_{n\to+\infty}(u_n - an - b) = 0$. Soit $(v_n)_{n\geq 0}$ la suite définie par : $v_n = u_n - an - b$, déterminez une fonction f telle que : $f(v_n) = \frac{1}{an+b}$ et, sans essayer de la calculer, utilisez f^{-1} pour en déduire un développement asymptotique de u_n en fonction de n.

Comment peut-on en déduire une formule approchée de u_n ? (N'oubliez pas d'étudier l'erreur).

5. Soit : $f(x) = 0$ une équation de racine x_0, déterminez φ de la forme :

$$\varphi(x) = x + \lambda(x)f(x) + \mu(x)f^2(x) \text{ telle que : } \varphi'(x_0) = \varphi''(x_0) = 0.$$

Programmez cette méthode, comparez-la avec la méthode de Newton. Généralisez.

6. Soit f une fonction de classe C^2 sur [a,b] telle que : $f(a)f(b) < 0$, f' et f" positifs strictement pour tout x de [a,b]. Soit $(u_n)_{n\geqslant 0}$ la suite définie par :

$$\begin{cases} u_0 = b \quad \text{et} \\ u_{n+1} = u_n - \dfrac{f(u_n)}{f'(u_n)} \text{ pour tout } n \geqslant 0. \end{cases}$$

Montrez que $(u_n)_{n\geqslant 0}$ est décroissante et minorée, déduisez-en qu'elle converge vers la racine x_0.

7. Soit φ : [a,b] $\longrightarrow$ [a,b] une fonction continue sur [a,b], montrer que φ admet un point fixe. Est-il unique ?

Evaluation asymptotique de la vitesse de convergence.

8. Soit φ : [a,b] $\longrightarrow$ [a,b] une fonction de classe C^2 sur [a,b] admettant un seul point fixe x_0 sur [a,b].

a. Si $|\varphi'(x_0)| > 1$, montrez que x_0 est répulsif.

b. Si $|\varphi'(x_0)| < 1$, montrez que x_0 est attractif et que si $(u_n)_{n\geqslant 0}$ est une suite définie par :

$$\begin{cases} u_0 \text{ "assez proche" de } x_0 \quad \text{et} \\ u_{n+1} = \varphi(u_n) \text{ pour tout } n \geqslant 0 \end{cases}$$

- si $\varphi'(x_0) \neq 0$, alors :

$$|u_{n+1} - x_0| \underset{n\to+\infty}{\sim} |\varphi'(x_0)|\ |u_n - x_0|$$

et la convergence est dite d'ordre un (ou linéaire).

- si $\varphi'(x_0) = 0$ et $\varphi''(x_0) \neq 0$, alors :

$$|u_{n+1} - x_0| \underset{n\to+\infty}{\sim} \frac{|\varphi''(x_0)|}{2}\ |u_n - x_0|^2$$

et la convergence est dite d'ordre deux (ou quadratique).

- Généralisez.

c. Si $|\varphi'(x_0)| = 1$, montrez que le point peut être répulsif ou attractif (voir également l'exercice 12).

9. Appliquez les idées de l'exercice 8 au cas de la méthode de Newton.

10. Dans cet exercice, on reprend - pour les préciser - les études des exercices 8 et 9.

a. Dans le cas $0 < |\varphi'(x_0)| < 1$, on peut penser qu'il existe un réel $m > 0$ tel que $|u_n - x_0| \underset{n\to+\infty}{\sim} m|\varphi'(x_0)|^n$.

Démontrez-le en considérant la suite $(v_n)_{n\geqslant 0}$ de terme général : $v_n = \dfrac{|u_n - x_0|}{|\varphi'(x_0)|^n}$ et la série $\sum \operatorname{Log} \dfrac{v_{n+1}}{v_n}$.

b. Dans le cas $\varphi'(x_0) = 0$ et $\varphi''(x_0) \neq 0$, montrez de même qu'il existe $0 < k < 1$ tel que : $|u_n - x_0| \underset{n\to+\infty}{\sim} \dfrac{2}{|\varphi''(x_0)|} k^{(2^n)}$.

c. Généralisez.

d. Appliquez ce qui précède au cas de la méthode de Newton.

11. On se place dans les hypothèses de l'exercice 6. Montrez que :

$$|u_{n+1}-x_0| \underset{n\to+\infty}{\sim} \frac{1}{2}\left|\frac{f''(x_0)}{f'(x_0)}\right| |u_n-x_0|^2 \text{ et que :}$$

$$|u_{n+1}-x_0| \leqslant k|u_n-x_0|^2 \text{ pour tout } n\geqslant 0, \text{ avec :}$$

$$k = \frac{1}{2}\frac{\sup\limits_{x\in[a,b]} |f''(x)|}{\inf\limits_{x\in[a,b]} |f'(x)|}.$$

12. Soit $(u_n)_{n\geqslant 0}$ la suite définie par :

$$\begin{cases} u_0 = a \in \mathbb{R} \text{ et} \\ u_{n+1} = \sin u_n \text{ pour tout } n\geqslant 0 \end{cases}$$

Etudiez sa convergence. Déterminez un réel α tel que la suite $\left(u_{n+1}^{\alpha}-u_n^{\alpha}\right)_{n\geqslant 0}$ admette admette une limite finie non nulle. Déduisez-en la partie principale de $(u_n)_{n\geqslant 0}$.

Que pensez-vous de la vitesse de convergence ?

Généralisez au cas $\varphi'(x_0) = 1$ de l'exercice 10.

13. Soit l'équation de Kepler : $u - 0{,}2 \sin u = 134°$. Utilisez la méthode des approximations successives pour la résoudre en calculant en radians puis en degrés. Comparez. Expliquez.

Justifiez le nombre d'itérations du programme du paragraphe 4.C.b.

14. Programmez les calculs arithmétiques (additions, multiplications, divisions, extractions de racines) en double, triple, … , précision.

15. Programmez les éphémérides de la Lune (voir le DANJON). Utilisez ce programme pour prévoir les éclipses.

16. *Méthode d'Aitken*

Soit $(u_n)_{n\geqslant 0}$ une suite donc la convergence est linéaire (voir l'exercice 8) et Δ l'endomorphisme de l'espace des suites définies par : $\Delta u_n = u_{n+1}-u_n$.

Soit $(v_n)_{n\geqslant 0}$ la suite définie par : $v_n = u_n - \dfrac{(\Delta u_n)^2}{\Delta^2 u_n}$ ($\Delta^2 = \Delta \circ \Delta$), montrez que $(v_n)_{n\geqslant 0}$ converge plus vite que $(u_n)_{n\geqslant 0}$. Programmez cette méthode. Comparez.

17. Résolvez l'équation : $x^3 - 2x + 2 = 0$ en utilisant la méthode de Newton. Etudiez le comportement de la suite (u_n) ainsi définie quand le point de départ est : $a = 1$ puis quand a est "proche" de 1. Plus généralement, étudiez le comportement de cette suite suivant le choix de a.

18. Reprenez l'exercice 17 pour une équation du troisième degré quelconque. Etudiez les différents cas possibles.

19. Quels sont les invariants des boucles des programmes de ce chapitre ?

CHAPITRE 9

LA RÉSOLUTION DES ÉQUATIONS ALGÉBRIQUES

Les équations algébriques, c'est-à-dire de la forme : $f(x) = 0$ où f est un polynôme, peuvent se résoudre par les méthodes des chapitres 7 et 8. Dans ce cas, il est nécessaire de commencer par séparer les racines. Dans ce chapitre, nous voyons une méthode qui donne directement les racines : *la méthode de Bairstow*. Elle consiste à décomposer le polynôme en produit de polynômes du second degré.

1. METHODE DE BAIRSTOW

A. PRINCIPES

Soit $f(x) = \sum_{i=o}^{n} a_i x^{n-i}$ le polynôme et (p,q) deux nombres, la division de $f(x)$ par $x^2 + px + q$ donne :

$$\sum_{i=o}^{n} a_i x^{n-i} = \left(\sum_{i=2}^{n} b_i x^{n-i} \right)(x^2+px+q) + rx + s$$

Les coefficients b_i, r et s s'obtiennent à partir de p et q par :

$$b_0 = b_1 = 0 \quad \text{et} \quad (1)\ b_i = a_{i-2} - pb_{i-1} - qb_{i-2} \text{ pour } 2 \leqslant i \leqslant n+2$$

alors : $$(2)\ r = b_{n+1} \text{ et } s = b_{n+2} + pb_{n+1}$$

comme on l'a vu au chapitre 3.

Les coefficients b_i, r et s sont des fonctions de (p,q), on cherche (p,q) tel que :

$$r(p,q) = s(p,q) = 0.$$

La méthode de Newton nous permet de déterminer une racine de ce système d'équations. En effet, par le calcul différentiel, il se linéarise en :

$$(3)\quad \begin{cases} r + \Delta p \dfrac{\partial r}{\partial p} + \Delta q \dfrac{\partial r}{\partial q} = 0 \\ s + \Delta p \dfrac{\partial s}{\partial p} + \Delta q \dfrac{\partial s}{\partial q} = 0 \end{cases}$$

Donc, après avoir essayé (p,q), nous essayons (p+Δp,q+Δq) où (Δp,Δq) est solution du système linéaire (3)).

Pour calculer $\frac{\partial r}{\partial p}$, il suffit de calculer les $\frac{\partial b_i}{\partial p}$ par récurrence, en dérivant la formule (1) :

$$\frac{\partial b_i}{\partial p} = -b_{i-1} - p\frac{\partial b_{i-1}}{\partial p} - q\frac{\partial b_{i-2}}{\partial p}.$$

Donc, si je pose : $c_i = \frac{\partial b_i}{\partial p}$ pour : $0 \leqslant i \leqslant n+2$, la suite (c_i) est définie par les formules :

$$c_0 = c_1 = 0 \quad \text{et} \quad c_i = -b_{i-1} - pc_{i-1} - qc_{i-2} \text{ pour } 2 \leqslant i \leqslant n+2.$$

Alors, d'après les formules (2) :

$$\frac{\partial r}{\partial p} = c_{n+1} \quad \text{et} \quad \frac{\partial s}{\partial p} = -qc_n.$$

D'autre part, en dérivant les formules (1), vous remarquerez que : $\frac{\partial b_i}{\partial q} = c_{i-1}$ pour tout i, donc :

$$\frac{\partial r}{\partial q} = c_n \quad \text{et} \quad \frac{\partial s}{\partial q} = c_{n+1} + pc_n.$$

Le système (3) se résout alors à l'aide des formules de Cramer :

$$d = c_{n+1}^2 - c_n(c_{n+2} + b_{n+1})$$

$$\Delta p = \frac{c_n b_{n+2} - b_{n+1}c_{n+1}}{d} \quad \text{et} \quad \Delta q = \frac{-b_{n+1}(qc_n + pc_{n+1}) - b_{n+2}c_{n+1}}{d}.$$

Nous définissons ainsi une suite de points (p,q) qui, en principe, converge vers un point (p,q) tel que : $x^2 + px + q$ divise f(x). Pratiquement, on limite le nombre d'itérations (à K = 100 dans le programme que je donne) et on utilise comme test d'arrêt :

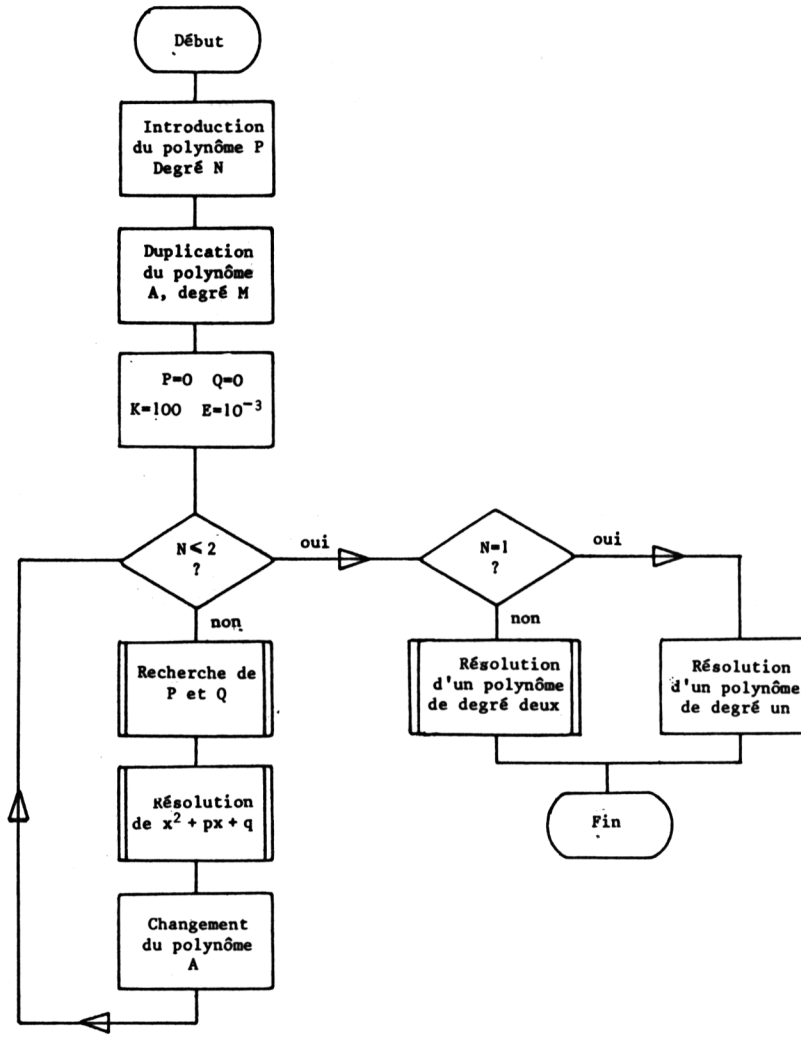
$$\frac{|\Delta p| + |\Delta q|}{|p| + |q|} < \varepsilon$$

ε étant donné à l'avance (10^{-3} dans le programme).

L'équation : $x^2 + px + q = 0$ est ensuite résolue sur ℂ par les formules usuelles, une première indication de l'erreur sur la valeur x_0 trouvée est donnée par : $\left|\frac{f(x_0)}{f'(x_0)}\right|$ (d'après la formule de Taylor).

Le programme continu ensuite avec le nouveau polynôme $\sum_{i=2}^{n} b_i x^{n-i}$ si son degré est supérieur à 3.

B. LOGIQUES DE DEROULEMENT DU PROGRAMME



Le programme exige donc plusieurs modules de programme :

- recherche de p et q par la méthode décrite précèdemment.
- résolution d'une équation du second degré.
- calcul de $\left|\frac{f}{f'}\right|$ avec l'algorithme de Hörner (voir le chapitre 3) modifié pour faire les calculs sur des nombres complexes.

C. LISTE DU PROGRAMME

```
 10 ! Resolution des equations
 20 !      algebriques
 30 !     *************
 40 !
 50 ! Introduction du polynome
 60 !       Σ A(I)X^N-I
 70 !     et duplication
 80 !
 90 INPUT N@ M=N
100 FOR I=0 TO N
110 PRINT "A(";I;")=";
120 INPUT A(I)@ P(I)=A(I)
130 NEXT I
140 !     Initialisation
150 P=0 @ Q=0
160 K=100 @ E=.001
170 !     Test du degre
180 ! Boucle de factorisation
190 ! ***********************
200 !
210 IF N<=2 THEN 570
220 !    Recherche de P et Q
230 !      ***************
240 !    Mise du compteur de
250 !     boucles J a zero
260 J=0
270 ! Test du nombre de boucles
280 IF J>K THEN 990
290 J=J+1
300 !  Calcul des suites B(I)
310 !         et C(I)
320 B(0)=0 @ B(1)=0 @ C(0)=0 @ C
    (1)=0
330 FOR I=2 TO N+2
340 B(I)=A(I-2)-P*B(I-1)-Q*B(I-2
    )
350 C(I)=-B(I-1)-P*C(I-1)-Q*C(I-
    2)
360 NEXT I
370 !    Calcul de ΔP et ΔQ
380 !
390 X=B(N+1) @ Y=B(N+2) @ Z=C(N)
     @ T=C(N+1) @ U=C(N+2)
400 D=T*T-Z*(U+X)
410 IF D=0 THEN 990
420 A=(Z*Y-X*T)/D
430 B=(-X*(Q*Z+P*T)-Y*T)/D
440 !     Nouveaux P et Q
450 P=P+A @ Q=Q+B
460 F=(ABS(A)+ABS(B))/(ABS(P)+AB
    S(Q))
470 IF F>E THEN 280
480 !  Le facteur est trouve
490 GOSUB 670
500 !  Changement de polynome
510 !   ********************
520 N=N-2
530 FOR I=0 TO N
540 A(I)=B(I+2)
550 NEXT I
560 GOTO 170
570 !     Dernier facteur
580 !       ***********
590 IF N=2 THEN 630
600 X=-A(1)/A(0) @ Y=0
610 GOSUB 800 ! Sortie
620 GOTO 650
630 P=A(1)/A(0) @ Q=A(2)/A(0)
640 GOSUB 670
650 END
660 !
670 ! Resolution de X^2+PX+Q
680 !    *****************
690 D=P*P-4*Q
700 IF D<0 THEN 750
710 D=SQR(D) @ Y=0
720 X=(-P+D)/2 @ GOSUB 800
730 X=(-P-D)/2 @ GOSUB 800
740 RETURN
750 D=SQR(-D)/2 @ X=-P/2
760 Y=D @ GOSUB 800
770 Y=-D @ GOSUB 800
780 RETURN
790 !
800 !     Sortie du resultat
810 !       ***********
820 PRINT X;Y
830 !     Calcul de l'erreur
840 !       (Approximation)
850 U=P(0) @ V=0
860 FOR I=1 TO M
870 R=U*X-V*Y @ V=U*Y+V*X @ U=R+
    P(I)
880 NEXT I
890 A=SQR(U*U+V*V)
900 U=0 @ V=0
910 FOR I=1 TO M
920 R=U*X-V*Y @ V=U*Y+V*X @ U=R+
    (M-I+1)*P(I-1)
930 NEXT I
940 IF A=0 THEN 960
950 A=A/SQR(U*U+V*V)
960 PRINT A
970 RETURN
980 !
990 PRINT "Processus divergent"
1000 END
```

Ce programme peut être simplifié en supprimant le calcul d'une approximation de l'erreur :

- suppression de la duplication (lignes 90 et 120).
- suppression des lignes 830 à 960.

D. UTILISATION

a. Soit à résoudre l'équation :

$$x^6 - 127x^5 + 215x^4 + 28x^3 - 39x^2 + 20x - 15 = 0 \ .$$

L'exécution du programme donne rapidement :

racines	indication d'erreur
0,03989619444 + 0,4466717900i	10^{-11}
0,03989619444 - 0,4466717900i	10^{-11}
0,5238337918	10^{-6}
- 0,6457525530	10^{-5}
125,2821089	10^{-8}
1,760014124	10^{-6}

Pour certaines racines, il est peu probable que tous les chiffres donnés soient significatifs.

Il est simple de vérifier la précision sur les racines réelles en calculant des valeurs du polynôme : pour cela j'utilise l'algorithme de Hörner en calculant soigneusement l'incertitude sur le résultat (voir le paragraphe 4.B. du chapitre 6 et le paragraphe 2.A. du chapitre 7). Ce qui donne pour la première racine réelle :

x	f(x)	Δf
0,523833	$- 1{,}6 \times 10^{-4}$	5×10^{-9}
0,523834	$- 8{,}5 \times 10^{-5}$	5×10^{-9}
0,523835	$- 7{,}0 \times 10^{-6}$	5×10^{-9}
0,523836	$7{,}1 \times 10^{-5}$	5×10^{-9}

Donc cette racine est : 0,523835 à 10^{-6} près.

Si vous voulez une meilleure précision, il vous suffit alors d'utiliser la méthode de Newton (vous pouvez également utiliser ce programme avec $E = 10^{-8}$).

De même pour les autres racines réelles :

$-0{,}64575$ à 10^{-5} près ; 125,2821089 à 10^{-7} près ; 1,760013 à 10^{-6} près.

Pour les racines complexes, cela est plus délicat, nous pouvons penser qu'elles sont :

$$0{,}0398962 \pm 0{,}4466718i \text{ à } 10^{-7} \text{ près.}$$

Nous reviendrons sur ce point au paragraphe 2.

b. *Cas de divergence*

Le processus a tendance à diverger si le polynôme a beaucoup de coefficients nuls. Dans ce cas, il suffit souvent de changer les valeurs initiales de P et Q (ligne 150 du programme) pour qu'il converge.

Soit par exemple, à résoudre l'équation : $x^3 - 3x + 1 = 0$. L'exécution du programme donne : "Processus divergent". En changeant la ligne 150 par :

150 P=1 : Q=1

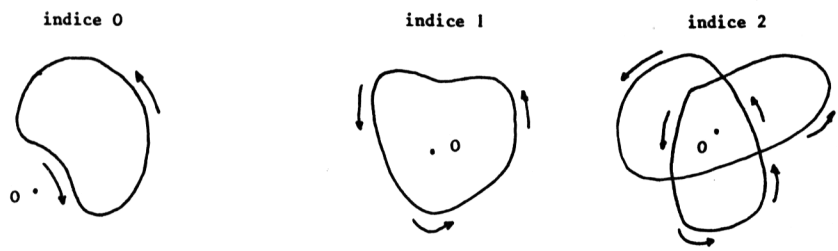
on obtient le résultat : 0,34730 -1,87938 1,53208 à 10^{-5} près.

2. PRECISION DES RACINES IMAGINAIRES

A. PRINCIPE

Pour vérifier si x_0 est valeur approchée avec une incertitude ε d'une racine de l'équation : $f(x) = 0$, il faut s'assurer que le disque D_ε d'équation : $|x-x_0| \leq \varepsilon$ contient une racine de : $f(x) = 0$, c'est-à-dire que $f(D_\varepsilon)$ (l'image de D_ε par f) contient 0.

Pour vérifier si $0 \in f(D_\varepsilon)$, le plus simple est de calculer l'indice de la frontière orientée de $f(D_\varepsilon)$ par rapport à 0, c'est-à-dire le nombre de fois que cette courbe entoure le point 0 dans le sens positif.



(Cette notion est précisée dans l'exercice 9).

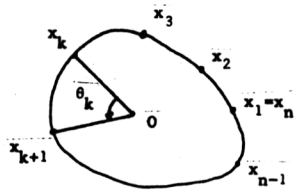
On démontre (voir l'exercice 10) que l'indice de la frontière de $f(D_\varepsilon)$ orientée positivement par rapport à 0 est égal au nombre de racines comptées avec leurs ordres de multiplicité contenues dans D_ε.

Pour calculer une valeur approchée de cet indice, je choisis des points $x_1, x_2, \ldots, x_n$ (avec $x_n = x_1$) de la courbe dans le sens de son orientation ; si θ_k est l'angle $(\overrightarrow{0x_k}, \overrightarrow{0x_{k+1}})$ alors l'indice est :

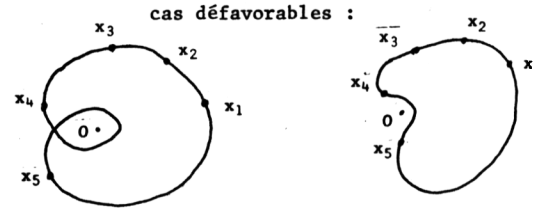
$$\frac{1}{2\pi}\sum_{k=1}^{n-1}\theta_k$$

à condition toutefois qu'une boucle entière ne soit pas intercalée entre deux points consécutifs :

cas usuel :



cas défavorables :



θ_k est l'argument du nombre $x_k x_{k+1}$, il est donc simple à calculer.

La frontière orientée de $f(D_\varepsilon)$ a pour équation paramétrique : $x = f(x_0+\varepsilon e^{it})$ où t décrit l'intervalle $[0,2\pi]$, les points x_k correspondent donc à une subdivision de $[0,2\pi]$.

B. PROGRAMME

Voici un programme qui réalise ce calcul :

```
10 ! Verification d'une racine
20 !         complexe
30 C=COS(PI/10) @ S=SIN(PI/10)
40 ! Introduction du polynome
50 INPUT N
60 FOR I=0 TO N
70 PRINT "A(";I;")=";
80 INPUT A(I)
90 NEXT I
100 ! Introduction de la racine
110 !    et de la precision
120 INPUT A,B,E
130 X=E @ Y=0 @ T=0
140 GOSUB 240
150 FOR I=1 TO 20
160 L=U @ M=V
170 R=C*X-S*Y @ Y=S*X+C*Y @ X=R
180 GOSUB 240
190 T=T+ATN2(V*L-U*M,U*L+V*M)
200 NEXT I
210 T=T/(2*PI)
220 PRINT T
230 GOTO 100
240 W=A+X @ Z=B+Y
250 U=A(0) @ V=0
260 FOR J=1 TO N
270 R=U*W-V*Z @ V=U*Z+V*W @ U=R+
    A(J)
280 NEXT J
290 RETURN
```

J'ai subdivisé l'intervalle $[0,2\pi]$ en vingt parties égales. Les valeurs successives de εe^{it} ont été calculées itérativement (rotation de centre 0 et d'angle $\frac{\pi}{10}$). Le sens de l'instruction ATN2 a été vu au paragraphe 4.C.b. du chapitre 8.

C. UTILISATION

Reprenons notre exemple, pour 0,03989619444 + 0,4466717900i et $\varepsilon = 10^{-11}$, l'exécution du programme donne : $1{,}3 \times 10^{-13}$ ce qui laisse penser que la précision obtenue n'est pas 10^{-11}.

Par contre, pour 0,0398961944 + 0,4466717900i et $\varepsilon = 10^{-10}$, on obtient
! d'où le résultat.

Remarquez qu'ici je n'ai pas une certitude totale car je n'ai pas étudié l'incertitude sur le calcul de l'indice.

D. PRECISION

Si on a besoin d'assurer une grande précision sur des racines imaginaires, le plus simple est d'utiliser les deux programmes qui précèdent pour localiser les racines puis la méthode de Newton en faisant un calcul d'erreur précis grâce au théorème du point fixe (voir le paragraphe 1.B.b du chapitre 8).

3. RACINES MULTIPLES

La méthode de Bairstow ne permet jamais d'affirmer qu'une racine est multiple même si elle permet de le soupçonner.

Prenons l'exemple de l'équation :

$$70x^5 - 99x^4 - 280x^3 + 396x^2 + 280x - 396 = 0.$$

L'exécution de notre programme donne : 1,414 - 1,414 1,416 - 1,416 1,414 à 10^{-3} près. Une vérification montre que l'incertitude sur la troisième et la quatrième est mauvaise. Je suis amené à conjecturer que 1,414 est racine triple et - 1,414 double. Pour le vérifier, un calcul approché est inopérant, la seule méthode est de calculer le p.g.c.d. de f et f', je trouve en utilisant l'algorithme d'Euclide :

$$\text{p.g.c.d. } (f,f') = x^2 - 2$$

donc $\sqrt{2}$ et $-\sqrt{2}$ sont racines doubles. En divisant f(x) par $(x^2-2)^2$, j'obtiens :

$$f(x) = (x^2-2)^2(70x-99).$$

Les racines sont donc $\sqrt{2}$ et $-\sqrt{2}$ doubles et $\frac{99}{70}$.

4. EXERCICES

1. Résolvez les équations :

$$x^{10} - 56x^9 + 48x^8 + 125x^7 - 7x^6 - 228x^5 + 25x^4 + x^3 - 2x^2 + x + 1 = 0$$
$$x^9 - 26x^6 + 7x^4 - 3x^3 + 2x^2 - 1 = 0$$
$$x^{15} - 27x^5 + 12x^3 - 56x + 21 = 0$$
$$2x^5 - 100x^2 + 2x - 1 = 0$$
$$x^4 - 4x + 1 = 0$$

Dans chaque cas, vous déterminerez soigneusement la précision des résultats.

Effet d'une incertitude sur les coefficients

2. Comparez les racines des équations

$$x^3 + 3x^2 + 2x - 1 = 0 \qquad x^3 + 3x^2 + 2x - 1{,}001 = 0$$
$$x^3 + 3{,}001x^2 + 2x - 1 = 0 \qquad x^3 + 3x^2 + 1{,}999x - 1 = 0 \quad \text{etc ...}$$

(Vous constaterez que la méthode de Bairstow diverge pour P = Q = 0, prenez par exemple P = Q = 2).

3. Soit $f : \mathbb{R}^4 \to \mathbb{R}$ définie par : $f(x,a,b,c) = x^3 + ax^2 + bx + c$, et x_0 une racine de $x^3 + 3x^2 + 2x - 1 = 0$. Montrez qu'il existe un voisinage V de (3,2,-1) et $\varphi : V \to \mathbb{R}$ de classe C^∞ telle que : $\varphi(3,2,-1) = x_0$ et $\varphi(a,b,c)$ racine de l'équation : $x^3 + ax^2 + bx + c = 0$. Calculez les dérivées de φ en (3,2,-1), déduisez-en une évaluation de l'erreur sur les racines de : $x^3 + 3x^2 + 2x - 1 = 0$ si les coefficients sont connus avec une incertitude de 10^{-3}.

4. Etudiez de même les équations de l'exercice 1 et l'équation : $x^3 - 3x + 2 = 0$.

5. En utilisant le programme précédent, créez un logiciel de décomposition des fractions rationnelles en éléments simples puis un logiciel de primitivation des fractions rationnelles.

6. *Localisation des racines*

Soit l'équation : $\sum_{i=o}^{n} a_i x^{n-i} = 0$. Montrez que les racines sont toutes contenues dans le disque de centre 0 et de rayon : $1 + \dfrac{\underset{1 \leqslant i \leqslant n}{\text{Max}} |a_i|}{|a_0|}$. De plus, si $a_n \neq 0$, les racines sont à l'extérieur du disque de centre 0 et de rayon l'inverse de $1 + \dfrac{\underset{0 \leqslant i \leqslant n-1}{\text{Max}} |a_i|}{|a_n|}$.

Ce résultat peut être utile si la méthode de Bairstow diverge (après plusieurs essais des valeurs P et Q).

7. *Méthode de Graeffe*

Nous considérons l'équation : (1) $\sum_{i=o}^{n} a_i x^{n-i} = 0$ de racines $x_1, x_2, \ldots, x_n$.

a. Supposons que les racines de l'équation (1) soient très étagées en module, c'est-à-dire que : $|x_1| \geqslant |x_2| \geqslant |x_3| \geqslant \ldots \geqslant |x_n|$.

Montrez alors que les racines sont approximativement : $-\dfrac{a_1}{a_0}, -\dfrac{a_2}{a_1}, \ldots, -\dfrac{a_n}{a_{n-1}}$.
Le principe de la méthode de Graeffe est de se ramener à ce cas.

b. Etant donné l'équation (1), formez l'équation (2), dont les racines sont : $-x_1^2, -x_2^2, \ldots, -x_n^2$. Constatez qu'en répétant cette opération plusieurs fois, si l'équation (1) n'a pas de racines de modules égaux, nous nous ramenons à l'hypothèse de a.

c. Il est d'usage d'arrêter les itérations quand les coefficients du nouveau polynôme sont - dans la limite de la précision des calculs - égaux aux carrés des coefficients du polynôme précédent. Expliquez pourquoi?

d. Programmez cette méthode. Quelle est son domaine d'application ? Comparez avec la méthode de Bairstow.

8. *Généralisation de la méthode de Graeffe*
Le contexte est celui de l'exercice 7.

a. Déterminez les itérées du type (2) associées à l'équation : $x^4 + x^3 - 10x^2 - 34x - 26 = 0$. Remarquez la variation des signes des coefficients.

b. Remarquez que la suite des signes des coefficients des itérées permet de déterminer la place des racines complexes.

c. Déduisez-en le module puis l'argument des racines complexes conjuguées.

d. Programmez cette méthode.

9. *Indice d'une courbe*
Soit Γ une courbe fermée orientée ne passant pas par 0 de paramètrage admissible : $[a,b] \to \mathbb{C}$, $t \to x(t)$ où x est une fonction de classe C^1.

a. Soit $f : [a,b] \to \mathbb{C}$ définie par : $f(t) = \exp\left[-\int_a^t \frac{x'(\tau)d\tau}{x(\tau)}\right] x(t)$. Montrez que f est constante. Déduisez-en que :

$$I = \frac{1}{2i\pi} \int_a^b \frac{x'(t)}{x(t)} dt \text{ est un entier.}$$

b. Montrez que : $\exp\left[\int_a^t \frac{x'(\tau)d\tau}{x(\tau)}\right] = \frac{x(t)}{x(a)}$ puis que : $\arg\left[\int_a^t \frac{x'(\tau)d\tau}{x(\tau)}\right] = \arg\left[\frac{x(t)}{x(a)}\right]$ pour tout t de [a,b].

c. Déduisez-en l'interprétation géométrique de I ainsi que le calcul approché proposé au paragraphe 2.A.

d. Montrez que I ne dépend pas du paramétrage admissible choisi. Que devient I par changement d'orientation ?

10. *Nombres de racines à l'intérieur d'un disque*
Nous considérons l'équation (1) : $f(x) = 0$ où f est un polynôme.

a. Soit $r > 0$ et $\theta \in \mathbb{R}$. Calculez : $\frac{1}{2\pi} \int_0^{2\pi} \frac{e^{it} dt}{e^{it} - re^{i\theta}}$.

b. Calculez $\frac{f'(x)}{f(x)}$ en fonction des racines de l'équation (1) et de leurs ordres de multiplicité.

c. Soit Γ le cercle de centre x et de rayon ε, calculez :

$$\frac{1}{2\pi} \int_0^{2\pi} \frac{f'(x+\varepsilon e^{it})\varepsilon e^{it}}{f(x+\varepsilon e^{it})} dt.$$

d. Démontrez le résultat du paragraphe 2.A.

11. Programmez la méthode de Newton pour le calcul des racines complexes des polynômes.

CINQUIEME PARTIE

LA RÉSOLUTION DES SYSTÈMES D'ÉQUATIONS

De même que pour les équations, la résolution des systèmes d'équations ne passe pas en général par une formule explicite de résolution. Par exemple dans le cas linéaire, les formules de Cramer se révèlent être la pire des méthodes envisageables pour résoudre un système.

Dans le chapitre 10, nous voyons la méthode de Gauss de résolution des systèmes linéaires et nous l'appliquons à des problèmes de circuits électriques ainsi qu'à des résolutions d'équations aux dérivées partielles (ce qui montre en outre l'intérêt des grands systèmes).

Dans le chapitre 11, la méthode de Newton (vue au chapitre 8) est appliquée à la résolution des systèmes non linéaires.

CHAPITRE 10

LA RÉSOLUTION DES SYSTÈMES D'ÉQUATIONS LINÉAIRES

Contrairement à une idée trop répandue, la résolution d'un système d'équations linéaires ne passe pas - sauf cas très particulier - par l'utilisation des formules de Cramer. Une méthode simple pour les petits systèmes (jusqu'à 100 équations environ) est la méthode de Gauss que je compare à la méthode de Cramer puis dont j'étudie la stabilité.

De plus la méthode de Gauss est également utile pour le *calcul d'un déterminant et de l'inverse d'une matrice.*

1. LA METHODE DE GAUSS

A. PRINCIPE

• En dehors des systèmes diagonaux, les systèmes les plus simples à résoudre sont, sans doute, les *systèmes triangulaires*. Par exemple, le système :

$$\begin{cases} x - 3y + 2z = 1 \\ 7y - 3z = -4 \\ 17z = 32 \end{cases}$$

se résoud très simplement en commençant par la troisième équation puis en portant le résultat obtenu dans la deuxième puis la première. D'où la solution : $x = -\frac{35}{17}$, $y = \frac{4}{17}$, $z = \frac{32}{17}$.

• L'idée de la méthode de Gauss est de se ramener à ce cas. Pour montrer comment, prenons l'exemple du système :

$$S \begin{cases} x - 3y + 2z = 1 & (1) \\ 2x + y + z = -2 & (2) \\ 3x - y + 5z = 3 & (3) \end{cases}$$

si (x,y,z) est solution de S, il est également solution de :

$$S' \begin{cases} x - 3y + 2z = 1 & (1') = (1) \\ 7y - 3z = -4 & (2') = (2) - 2(1) \\ 8y - z = 0 & (3') = (3) - 3(1) \end{cases}$$

car les équations de S' sont combinaisons linéaires des équations de S. La réciproque est également vraie car :

$$(1) = (1') \quad (2) = (2') + 2(1') \quad (3) = (3') + 3(1').$$

Donc (x,y,z) est solution de S si et seulement si il est solution de S' : on dit que S et S' sont *équivalents*.

De même S' est équivalent à :

$$S'' \begin{cases} x - 3y + 2z = 1 & (1'') = (1') \\ 7y - 3z = -4 & (2'') = (2') \\ 17z = 32 & (3'') = 7(3') - 8(2') \end{cases}$$

qui est triangulaire.

• Dans l'exemple précédent, nous sommes passés de S à S' en utilisant le coefficient de x de l'équation (1) comme *pivot* pour éliminer les coefficients de x des équations (2) et (3). Cette opération est possible à la condition que ce coefficient ne soit pas nul : une interversion des inconnues ou des équations permet de nous y ramener (si le système est de Cramer bien entendu).

B. DESCRIPTION DES CALCULS

a. *Pivotage*

Soit S le système de départ, je l'écris :

$$S \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1n}x_n = a_{1n+1} \\ a_{21}x_1 + a_{22}x_2 + \cdots\cdots + a_{2n}x_n = a_{2n+1} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots\cdots + a_{nn}x_n = a_{nn+1} \end{cases}$$

Le deuxième membre habituellement noté $b_1, b_2, \cdots, b_n$ étant noté a_{1n+1}, $a_{2n+1}, \cdots, a_{nn+1}$ par commodité.

Je définis ensuite une suite $S_0 = S$, S_1, S_2, ... , S_n de systèmes de façon que S_k soit de la forme :

$$S_k \begin{cases} a_{11}^{(k)} x_1 + a_{12}^{(k)} x_2 + \ldots\ldots\ldots\ldots + a_{1n}^{(k)} x_n = a_{1n+1}^{(k)} \\ a_{22}^{(k)} x_2 + \ldots\ldots\ldots\ldots + a_{2n}^{(k)} x_n = a_{2n+1}^{(k)} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{kk}^{(k)} x_k + \ldots\ldots + a_{kn}^{(k)} x_n = a_{k\,n+1}^{(k)} \\ a_{k+1\,k}^{(k)} x_k + \ldots\ldots + a_{k+1\,n}^{(k)} x_n = a_{k+1\,n+1}^{(k)} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{nk}^{(k)} x_k + \ldots\ldots + a_{nn}^{(k)} x_n = a_{n\,n+1}^{(k)} \end{cases}$$

Pour obtenir le système S_{k+1} à partir du système S_k, je cherche un pivot p_k parmi les coefficients $a_{ij}^{(k)}$ pour i et j variant de k à n. Le choix le plus logique est de prendre le coefficient maximal en valeur absolue (voir le chapitre 1 du tome 3 pour la recherche de celui-ci et l'exercice 6 pour la discussion de ce choix).

Si $p_k = 0$, c'est-à-dire si le déterminant du système S est nul, le calcul doit s'arrêter car le système n'est pas de Cramer ; du fait des erreurs de calcul, je teste plutôt : $|p_k| < \delta$, δ petit donné à l'avance (0,001 dans le programme).

J'échange alors la ligne k et la ligne du pivot, ainsi que la colonne k et la colonne du pivot. La ligne k est alors normalisée par division par p_k, le pivot devient donc égal à 1, je note S'_k le système obtenu et $a'^{(k)}_{ij}$ ses coefficients.

S_{k+1} est obtenu de S'_k par les opérations :

$$(1) \qquad a_{ij}^{(k+1)} = a'^{(k)}_{ij} - a'^{(k)}_{kj}\ a'^{(k)}_{ik}$$

pour i variant de k+1 à n et j de k+1 à n+1.

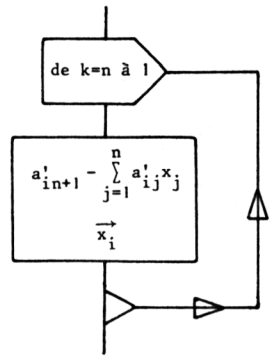
b. *Résolution*

Le système s'écrit alors :

$$S' \begin{cases} x_1 + a'_{12} x_2 + \ldots + a'_{1n} x_n = a'_{1\,n+1} \\ x_2 + \ldots + a'_{2n} x_n = a'_{2\,n+1} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ x_n = a'_{n\,n+1} \end{cases}$$

qui se résoud en commençant par la dernière équation.

Si les x_i sont initialisés à 0, l'algorithme :



fournit la solution.

C. COMPARAISON AVEC LA METHODE DE CRAMER

• Pour résoudre un système de n équations à n inconnues, la méthode de Cramer nécessite le calcul de n+1 déterminants d'ordre n.

D'après la définition : $\det(a_{ij}) = \sum \varepsilon(\sigma) a_{1\sigma(1)} \cdots a_{n\sigma(n)}$, la sommation étant étendue à l'ensemble des permutations σ de $\{1,2,\ldots,n\}$, $\varepsilon(\sigma)$ désignant la signature de σ, donc le calcul d'un déterminant d'ordre n nécessite $n \cdot n! - 1$ opérations. En tenant compte des divisions, la méthode de Cramer nécessite donc $(n+1)(n \cdot n! - 1) + n$ opérations.

• Pour sa $k^{\text{ième}}$ étape, la méthode de Gauss nécessite $(n-k)[2(n-k)+1]$ opérations donc le pivotage nécessite $\sum_{p=o}^{n-1} p(2p+1) = \frac{n(n-1)(4n+1)}{6}$ opérations (voir l'exercice 7).

$x_n, x_{n-1}, \cdots, x_{k+1}$ étant calculés, le calcul de x_k nécessite $2(n-k)$ opérations donc la résolution nécessite $2 \sum_{p=o}^{n-1} p = n(n-1)$ opérations. La méthode de Gauss nécessite donc $\frac{n(n-1)(4n+7)}{6}$ opérations.

• Vous pouvez remarquer l'écart énorme entre les temps de calcul. Pour un ordinateur de table exécutant 1000 opérations par seconde, cela fait 6 secondes avec la méthode de Gauss et 700 milliards d'années avec la méthode de Cramer pour résoudre un système 20×20.

D. PRECISION DES RESULTATS

De même que pour les équations numériques, je n'analyserai pas ici l'effet d'une incertitude sur les coefficients du système (voir cependant les exercices 9 et 10). Il est possible de faire une étude a posteriori de la pré-

cision comme nous l'avons fait pour les solutions réelles des équations numériques (voir les chapitres 7 et 8), elle consiste essentiellement à calculer les nombres : $a_{i1}x_1 + \cdots + a_{in}x_n - a_{in+1}$, $(x_1, \cdots, x_n)$ étant la solution présumée. Mais les erreurs de calculs sur ces nombres sont importantes et il est nécessaire de les connaître pour l'appliquer, c'est pourquoi je préfère exposer une méthode directe (cependant, l'exercice 11 reprend cette idée). En exercice 12, je propose également une méthode empirique d'étude de la précision.

Pour ce qui suit, j'ai choisi de suivre l'évolution des erreurs en constituant itérativement une matrice d'incertitudes Δ.

a. *Incertitude sur le pivotage*

Pour le calcul des incertitudes sur la résolution du système triangulaire, il importe que les coefficients nuls sous la diagonale et égaux à 1 sur la diagonale soient considérés comme exacts ce qui revient à considérer p_k et $a'^{(k)}_{kj}$ comme exacts dans la formule (1) du paragraphe 1.B.a (c'est-à-dire les remplacer par leurs valeurs approchées calculées).

Je note $\delta^{(k)}_{ij}$ l'incertitude sur $a^{(k)}_{ij}$ et $\delta'^{(k)}_{ij}$ celle sur $a'^{(k)}_{ij}$; les δ' sont obtenus des δ par la même transformation que les a' des a et :

$$\delta'^{(k)}_{kj} = \frac{\delta^{(k)}_{kj}}{|p_k|} + |a^{(k)}_{kj}|\varepsilon \text{ pour j variant de k+1 à n+1.}$$

Puis $\delta^{(k+1)}$ de $\delta'^{(k)}$ par :

$$\delta^{(k+1)}_{ij} = \delta'^{(k)}_{ij} + |a'^{(k)}_{kj}|\delta'^{(k)}_{ik} + |a^{(k+1)}_{ij}|\varepsilon$$

pour i variant de k+1 à n et j de k+1 à n+1, les autres étant inchangés.

ε est l'ordre de précision de l'ordinateur utilisé (10^{-10} dans le programme).

b. *Incertitude sur la résolution*

De même, je note Δx_i l'incertitude sur x_i, alors :

$$\Delta x_k = \delta^{(n)}_{k\,n+1} + \sum_{j=k+1}^{n}\left(|a'^{(n)}_{kj}|\Delta x_j + |x_j|\delta^{(n)}_{k,j}\right) + |x_k|\varepsilon$$

pour k variant de n à 1.

Le calcul peut donc se faire parallélement au calcul des x_i.

2. PROGRAMMATION

A. LISTE DU PROGRAMME

```
 10 !  Resolution d'un systeme
 20 !   d'equations lineaires
 30 !    ********************
 40 !
 50 ! Introduction des donnees
 60 !     et initialisation
 70 !       ************
 80 PRINT "Nombres d'inconnues";
 90 INPUT N
100 FOR I=1 TO N
110 FOR J=1 TO N
120 PRINT "A(";I;",";J;")";
130 INPUT A(I,J)
140 NEXT J
150 PRINT "B(";I;")";
160 INPUT A(I,N+1)
170 NEXT I
180 ! Valeur minimale pour le
190 !          pivot
200 D=.001
210 E=.0000000001
220 FOR I=1 TO N @ X(I)=0
230 Y(I)=0
240 FOR J=1 TO N+1 @ D(I,J)=0 @
    NEXT J
250 NEXT I
260 !        Pivotage
270 !       **********
280 !
290 FOR K=1 TO N
300 ! Determination du pivot P:
        Maximum des !A(I,J)!
           pour  I>=K
310 P=0
320 FOR I=K TO N
330 FOR J=1 TO N
340 IF ABS(A(I,J))>ABS(P) THEN L
    =I @ M=J @ P=A(I,J)
350 NEXT J
360 NEXT I
370 !       Test de P
380 IF ABS(P)<D THEN 820
390 ! Echange de la ligne du
      pivot et de la ligne K
400 FOR J=1 TO N+1
410 A=A(L,J) @ A(L,J)=A(K,J) @ A
    (K,J)=A
420 F=D(L,J) @ D(L,J)=D(K,J) @ D
    (K,J)=F
430 NEXT J
440 ! Numero de la colonne du
                pivot
450 A(K,0)=M
460 ! Normalisation de la ligne
              du pivot
470 FOR J=1 TO N+1
480 A(K,J)=A(K,J)/P
490 D(K,J)=D(K,J)/ABS(P)+ABS(A(K
    ,J))*E
500 NEXT J
510 ! Elimination de la colonne
              du pivot
520 FOR I=K+1 TO N
530 A=A(I,M)
540 F=D(I,M)
550 FOR J=1 TO N+1
560 A(I,J)=A(I,J)-A*A(K,J)
570 D(I,J)=D(I,J)+ABS(A(K,J))*F+
    ABS(A(I,J))*E
580 NEXT J
590 NEXT I
600 NEXT K
610 !       Resolution
620 !      ************
630 !
640 FOR K=N TO 1 STEP -1
650 X=A(K,N+1)
660 Y=D(K,N+1)
670 FOR J=1 TO N
680 X=X-A(K,J)*X(J)
690 Y=Y+ABS(A(K,J))*Y(J)+ABS(X(J
    ))*D(K,J)
700 NEXT J
710 X(A(K,0))=X
720 Y(A(K,0))=Y+X*E
730 NEXT K
740 !    Sortie des resultats
750 !  **********************
760 !
770 FOR J=1 TO N
780 PRINT "X(";J;")=";X(J)
790 PRINT "ΔX(";J;")=";Y(J)
800 NEXT J
810 GOTO 830
820 PRINT "Pivot < D"
830 END
```

La partie "calcul d'incertitude" de ce programme prend une place importante en mémoire, il est simple de la supprimer puisqu'elle constitue les lignes : 210, 230, 240, 420, 490, 540, 570, 660, 690, 720, 790.

B. UTILISATION DU PROGRAMME

a. Revenons sur l'exemple du paragraphe 1.A, l'exécution du programme précédent donne :

$$X(1) = -2{,}058823529 \qquad \Delta X(1) = 4 \times 10^{-10}$$
$$X(2) = 0{,}2352941176 \qquad \Delta X(2) = \ 10^{-10}$$
$$X(3) = 1{,}882352941 \qquad \Delta X(3) = 4 \times 10^{-10}$$

ce qui est bien le résultat attendu avec une précision de 10^{-9}.

b. Soit à résoudre le système :

$$\begin{cases} 5x + 7y + 6z + 5t = 23 \\ 7x + 10y + 8z + 7t = 32 \\ 6x + 8y + 10z + 9t = 33 \\ 5x + 7y + 9z + 10t = 31 \end{cases}$$

L'exécution du programme donne :

$$X(1) = 0{,}9999999997 \qquad \Delta X(1) = 2 \times 10^{-8}$$
$$X(2) = 1{,}000000001 \qquad \Delta X(2) = 2 \times 10^{-8}$$
$$X(3) = 1{,}000000001 \qquad \Delta X(3) = 5 \times 10^{-9}$$
$$X(4) = 0{,}9999999999 \qquad \Delta X(4) = 5 \times 10^{-9}$$

ce qui fait : $x = y = z = t = 1$ à 2×10^{-8} près mais une vérification rapide montre que cette valeur est exacte.

c. Stabilité

Une étude théorique de la stabilité de cet algorithme est possible mais, à l'heure actuelle, elle aboutit à des résultats beaucoup trop pessimistes par rapport à l'expérience (voir l'exercice 13 sur cette question). Par contre, une étude expérimentale vous montrera que la stabilité de la méthode de Gauss est excellente pour les petits systèmes.

La sortie peut être "pivot<D" bien que le système soit de Cramer mais dans ce cas la solution est vraisemblablement très peu stable par rapport aux erreurs de calcul comme aux erreurs dans les données (voir l'exercice 9).

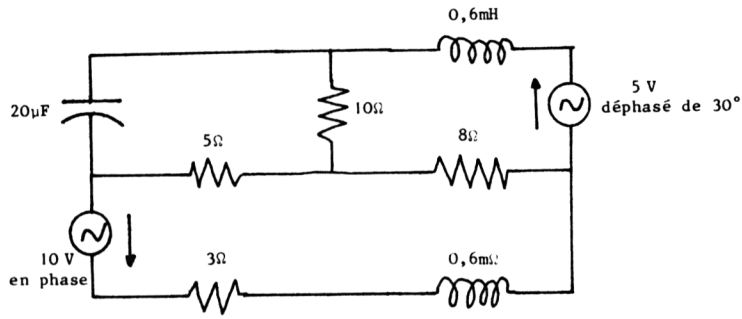
C. AMELIORATIONS

Il est facile de modifier ce programme pour qu'il fournisse le déterminant (au signe près, c'est le produit des pivots) et l'inverse d'une matrice : il suffit de résoudre n systèmes successivement (de second membre $(1,0,\ldots,0)$, $(0,1,0,\ldots,0)$, ... , $(0,\ldots,0,1)$). Il est également facile de le modifier pour résoudre de façon exacte des systèmes à coefficients rationnels, ou de façon approchée des systèmes à coefficients complexes. Une autre amélioration possible est la résolution de systèmes de n-1 équations à n inconnues, ou plus généralement de p équations ($p \leqslant n$) à n inconnues.

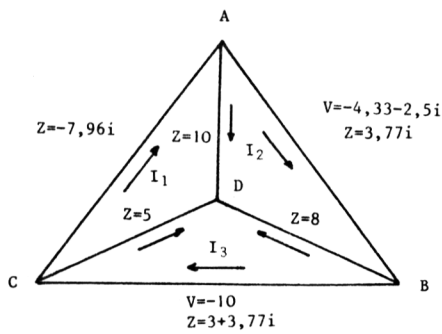
Ces diverses améliorations font l'objet des exercices 2 à 5.

3. CALCUL D'UN CIRCUIT ELECTRIQUE

Je considère le circuit suivant :



La fréquence est la même pour les deux sources de tension et vaut 1000 Hertzs. On désire calculer les courants qui circulent dans chaque branche du circuit. Pour cela, je schématise le circuit de façon plus simple, j'oriente les branches et je calcule leurs impédances :



La loi des noeuds appliquée en A,B et C donne les courants dans les branches AD, BD et CD : I_1-I_2, I_2-I_3 et I_3-I_1. La loi des mailles fournit alors :

pour ADC : $-7,96i\ I_1 + 10(I_1-I_2) + 5(I_1-I_3) = 0$

pour ABD : $3,77i\ I_2 + 8(I_2-I_3) + 10(I_2-I_1) = -4,33 - 2,5i$

pour BCD : $(3+3,77i)I_3 + 5(I_3-I_1) + 8(I_3-I_2) = -10$

ce qui donne le système :

$$\begin{cases} (15-7,96i)I_1 - 10\ I_2 - 5\ I_3 = 0 \\ -10\ I_1 + (18+3,77i)I_2 - 8\ I_3 = -4,33 - 2,5i \\ -5\ I_1 - 8\ I_2 + (16+3,77i)I_3 = -10 \end{cases}$$

qui se traduit en un système de 6 équations à 6 inconnues réelles. L'exécution du programme donne alors :

$$I_1 = -1,72 - 0,93i \quad I_2 = -2,21 - 0,12i \quad I_3 = -2,23 + 0,18i$$

c'est-à-dire $I_1 = 1,96$ A avec une avance de phase de 118°, de même pour les autres courants.

4. INTERET DES GRANDS SYSTEMES

Un des intérêts des grands systèmes est la résolution d'équations aux dérivées partielles, équations qui régissent bien des phénomènes physiques. Ce qui suit est une introduction aux méthodes numériques de résolution des équations aux dérivées partielles.

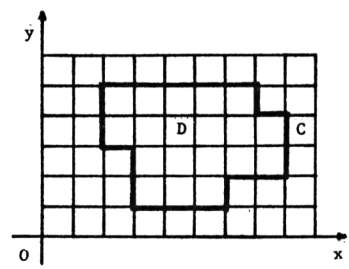
A. METHODE DES DIFFERENCES FINIES

Je considère un domaine D du plan limité par une courbe C et f une fonction définie sur C. Je cherche les fonctions u de classe C^2 sur D telles que :

$$\Delta u = 0 \text{ sur } D \quad \text{et} \quad u = f \text{ sur } C.$$

Ce problème est appelé "problème de Dirichlet" pour l'équation de Laplace (Δu est le laplacien de u).

Je me donne un *pas* $h > 0$ (h "petit"), je trace les droites d'équations : $x = ih$ et $y = jh$ pour (i,j) entiers et je suppose que D est la réunion d'un nombre fini des carrés ainsi définis (ce qui constitue une première approximation) :

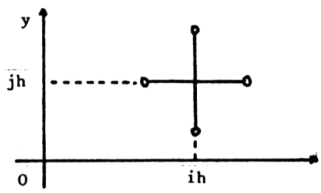


La méthode consiste à déterminer une valeur approchée de u aux *noeuds* de ce quadrillage (je note $u_{i,j} = u(ih,jh)$). Les noeuds se subdivisent en deux sous-ensembles :

- ceux qui appartiennent à la courbe C, en ces points :

$$u_{i,j} = f_{i,j} \text{ donnée;}$$

- ceux qui appartiennent à l'intérieur de D, en ces points :



D'après la formule de Taylor à l'ordre deux appliquée entre (ih,jh) et ((i+1)h,jh) :

$$u_{i+1,j} = u_{i,j} + h \frac{\partial u}{\partial x}_{i,j} + \frac{h^2}{2} \frac{\partial^2 u}{\partial x^2}_{i,j}$$

si on néglige le terme d'ordre trois ; de même en (i-1,j), (i,j+1), (i,j-1). Donc, en faisant la somme de ces quatre égalités :

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} = 4u_{i,j} + h^2 \Delta u_{i,j} = 4u_{i,j}$$

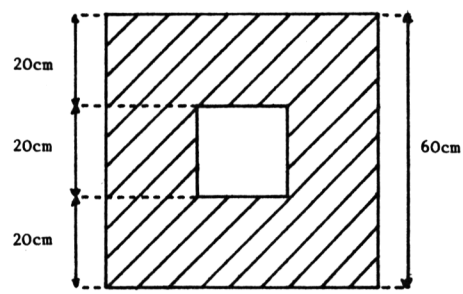
(ce qui constitue une seconde approximation).

Donc $(u_{i,j})$ vérifie un système de n équations à n inconnues si n est le nombre de noeuds inclus dans D (il est facile de montrer que ce système est de Cramer : voir l'exercice 14).

Il s'agit de résoudre ce système et vous voyez que l'on obtient ainsi naturellement de très grands systèmes (10000 équations à 10000 inconnues). Je ne discuterais pas la validité de cette méthode mais je vais en donner un exemple d'application :

B. REPARTITION DE LA TEMPERATURE DANS UNE PLAQUE

Je considère la plaque représentée par le dessin :

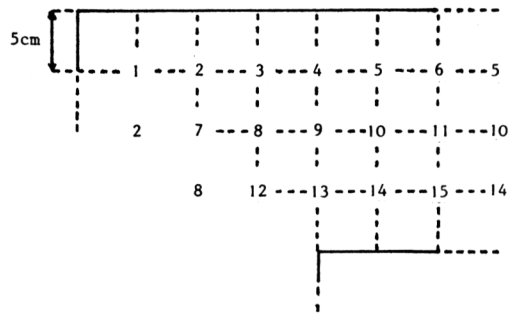


Le bord intérieur est maintenu à la température de 100° et le bord extérieur à 0° . Quelle est la répartition de température dans la plaque à l'état stationnaire ?

Si u(M) désigne la température au point M, u vérifie :

$\Delta u = 0$ à l'intérieur, $u = 100$ sur le bord intérieur et $u = 0$ sur le bord extérieur.

Il s'agit donc d'un problème de Dirichlet à résoudre ; en tenant compte des symétries, je le discrétise de la façon suivante :



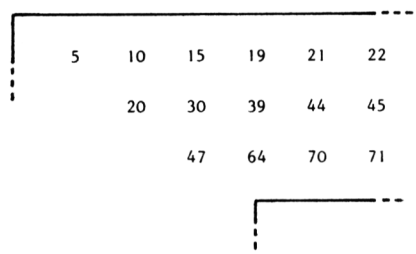
ce qui conduit à un système de 15 équations à 15 inconnues (voir la numérotation sur la figure), par exemple :

la première équation est : $4u_1 - 2u_2 = 0$

la huitième : $-u_3 - u_7 + 4u_8 - u_9 - u_{12} = 0$

la quatorzième : $-u_{10} - u_{13} + 4u_{14} - u_{15} = 100.$

J'obtiens la répartition :



En exercice 15, nous voyons une méthode mieux adaptée à ce genre de systèmes.

5. EXERCICES

1. Résolvez les systèmes :

$$\begin{cases} x + 2y + 3z + 4t = 7 \\ x + 5y + z + 9t = 29 \\ 3x + y + 8z - t = 20 \\ 4x - 2y + z + 7t = 1 \end{cases} \qquad \begin{cases} 2x + y + z + t = 1 \\ x + 2y + z + t = 0 \\ x + y + 2z + t = -1 \\ x + y + z + 2t = 2 \end{cases}$$

2. Programmez le calcul d'un déterminant, de l'inverse d'une matrice (utilisez les indications du paragraphe 2.C.).

3. Programmez la résolution exacte des systèmes d'équations linéaires à coefficients rationnels. Vous utiliserez la méthode de Gauss et les calculs sur les rationnels du chapitre 2. Reprenez l'exercice 1 avec ce programme.

Etendez ce programme aux calculs de déterminant et de l'inverse d'une matrice. Examinez le cas de coefficients dans un corps quadratique (voir l'exercice 5 du chapitre 2).

4. Programmez la résolution approchée des systèmes d'équations linéaires à coefficients complexes. Appliquez-la au calcul du circuit électrique du paragraphe 3.

Etendez ce programme aux calculs de déterminant et de l'inverse d'une matrice.

5. Programmez la résolution approchée des systèmes de p équations à n inconnues ($p \leqslant n$). Pour cela, vous utiliserez la méthode de Gauss.

Reprenez ceci dans les cas des exercices 3 et 4.

6. A l'étape k de la méthode de Gauss, il est possible de choisir comme pivot le coefficient de la ligne k maximal en valeur absolue. Comparez expérimentalement ce choix avec celui du paragraphe 1.B.a.

Faites de même en choisissant le coefficient de la colonne k sous la ligne k maximal en valeur absolue.

7. Montrez que : $\sum_{p=o}^{n-1} (p+1)^2 = \sum_{p=o}^{n-1} p^2 + 2\sum_{p=o}^{n-1} p + 2n$, déduisez-en $\sum_{p=o}^{n-1} p$ en fonction de n.

De même, en développant $(p+1)^3$, calculez $\sum_{p=o}^{n-1} p^2$. Déduisez-en $\sum_{p=o}^{n-1} p(2p+1)$.

8. *Etude d'une norme matricielle*

Soit B une matrice n×p (n et p entiers), on note : $\|B\| = \sqrt{\sum_{i,j} b_{i,j}^2}$ où (i,j) varie sur $\{1,2,\dots,n\} \times \{1,2,\dots,p\}$.

Soit A une matrice n×n, X et Y deux matrices n×1, montrez que :

a. $\|AX\| \leqslant \|A\| \cdot \|X\|$ et $\|X+Y\| \leqslant \|X\| + \|Y\|$

b. $\|AX\| \geqslant \sqrt{\lambda}\ \|X\|$ où λ est la plus petite valeur propre de la matrice ${}^tA.A$.

c. $\lambda \geqslant \dfrac{(\det A)^2}{\|A\|^{2(n-1)}} (n-1)^{n-1}$ (Vous remarquerez que $\|A\|^2 = \mathrm{tr}({}^tA.A)$)

d. $\|X\| \leqslant \dfrac{\|A\|^{n-1}}{(n-1)^{(n-1)/2}\,|\det A|}\ \|AX\|$.

Effet d'une incertitude sur les coefficients du système :

9. Comparez les solutions des systèmes :

$$\begin{cases} 0{,}999x + y = 0 \\ x + y = 1 \end{cases} \qquad \begin{cases} x + y = 0 \\ x + 1{,}001y = 1 \end{cases} \qquad \begin{cases} x + 0{,}999y = 0 \\ x + y = 1 \end{cases}$$

Créez des cas analogues de systèmes 3×3.

10. Soit AX = B l'écriture matricielle d'un système linéaire de n équations à n inconnues (A est une matrice n×n, B et X des matrices n×1). A et B sont connues avec des incertitudes ΔA et ΔB, c'est-à-dire que les valeurs données vérifient :

$$\|A-\overline{A}\| \leqslant \Delta A \quad \text{et} \quad \|B-\overline{B}\| \leqslant \Delta B$$

(voir l'exercice 8 pour la définition de la norme).

a. En supposant que les deux systèmes : AX = B et $\overline{A}X = \overline{B}$ sont de Cramer, et en notant X et $\overline{X}$ leurs solutions, montrez que :

$$\|X-\overline{X}\| \leqslant \frac{\|A\|^{n-1}}{(n-1)^{(n-1)/2}\,|\det A|} \left(\|\overline{X}\|\,\Delta A + \Delta B\right).$$

b. Appliquez ce qui précède aux systèmes de l'exercice 9 puis aux systèmes de l'exercice 1 en supposant que chaque coefficient est connu avec une incertitude de 10^{-3}.

11. *Etude a posteriori de l'erreur*

Soit $AX = B$ l'écriture matricielle d'un système (voir l'exercice 10) et $\overline{X}$ la solution trouvée.

a. Soit X la véritable solution, montrez que :

$$\|X-\overline{X}\| \leqslant \frac{\|A\overline{X}-B\|}{|\det A|} \frac{\|A\|^{n-1}}{(n-1)^{(n-1)/2}}$$

b. Utilisez cette méthode sans calcul d'erreur sur $A\overline{X}-B$, que constatez-vous ? Comment y remédier ?

c. Programmez un calcul reposant sur cette méthode.

12. Une méthode empirique d'étude de la précision des calculs est de résoudre successivement les systèmes $AX = B$ et $(\alpha A)X = (\alpha B)$ pour α réel proche de 1 ($\alpha = 1{,}369087542$ par exemple). Utilisez cette méthode, comparez expérimentalement ses résultats avec ceux des autres méthodes.

13. En utilisant les formules du paragraphe 1.D.a, montrez que l'erreur sur le pivotage est majorée par $\frac{\|A\|\varepsilon}{\delta} n\, 2^n$ où $\|A\| = \max_{i,j} |a_{i,j}|$ (i variant de 1 à n, j de 1 à n+1), ε la précision de l'ordinateur, δ le plus petit pivot (en valeur absolue) et n le nombre d'équations.

Déduisez-en l'erreur sur la résolution du système. Comparez expérimentalement avec les autres méthodes.

14. Le contexte de cet exercice est le paragraphe 4.A. Dans le cas $f = 0$, montrez, en considérant le maximum de $|u_{i,j}|$, que $u_{i,j} = 0$ pour tout (i,j). Déduisez-en que le système est de Cramer.

15. Le contexte de cet exercice est le paragraphe 4.A, $AX = B$ est l'écriture matricielle du système. Constatez que : $A = 4I - T$ où I est la matrice identité d'ordre n et T une matrice symétrique de diagonale nulle n'ayant que des 0 et des 1 (au maximum 4 par ligne et par colonne). Le système s'écrit donc : $X = \frac{1}{4}(B+TX)$.

a. Montrez que les valeurs propres de T sont de valeur absolue strictement inférieure à 4.

b. Déduisez-en que la suite (X_n) définie par : X_0 choisi arbitrairement et $X_{n+1} = \frac{1}{4}(B+TX_n)$ pour tout $n \geqslant 0$, converge vers la solution X du système. Evaluez $\|X-X_n\|$ en fonction de n.

c. Evaluez la taille mémoire minimale pour traiter ainsi un système de n équations ainsi que le nombre d'opérations par itération.

d. Programmez cette méthode. Vous utiliserez comme critère d'arrêt : $\frac{\|X_{n+1} - X_n\|}{\|X_n\|} \leqslant \varepsilon$, avec ε donné à l'avance. La norme étant : $\|X\| = \max_i |x_i|$.

CHAPITRE 11

LA RÉSOLUTION DES SYSTÈMES D'ÉQUATIONS NON LINEAIRES

Dans ce chapitre je montre que les idées du chapitre 8 s'appliquent aux systèmes d'équations, il ne contient donc aucun résultat théorique nouveau. En particulier, la précision des résultats peut être étudiée grâce au théorème du point fixe.

D'autre part, ce sujet sera repris au chapitre 2 du tome 3 avec le calcul du minimum d'une fonction de plusieurs variables.

1. ETUDE D'UN EXEMPLE

A. CALCUL

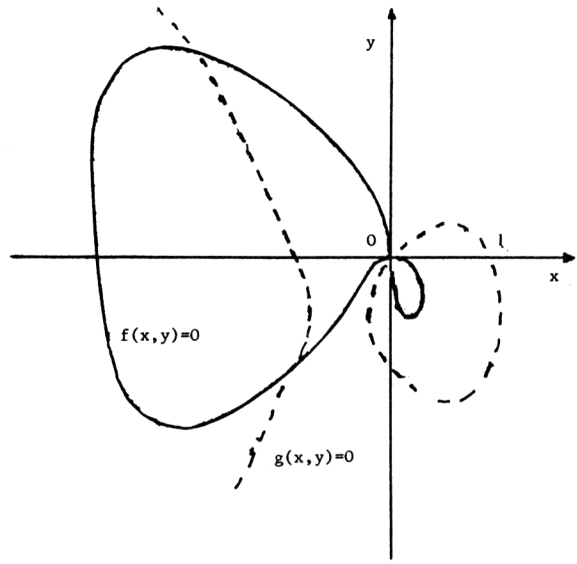
Soit à résoudre le système d'équations :

$$\begin{cases} x^4 + y^4 + 3x^3 + xy = 0 \\ x^3 + y^2 + y - x = 0. \end{cases}$$

D'après le tracé des courbes d'équation : $f(x,y) = x^4 + y^4 + 3x^3 + xy = 0$ et : $g(x,y) = x^3 + y^2 + y - x = 0$ (voir le tome 4) : le système a trois solutions : (0,0), (-1,-1) et une située près de (-2,2).

Pour le résoudre, je l'écris sous la forme : $\phi(X) = X$ en utilisant la méthode de Newton modifiée. Il s'écrit : $F(x) = 0$ avec :

$$X = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{et} \quad F(X) = \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$$



La matrice jacobienne de F est :

$$J_F(X) = \begin{pmatrix} 4x^3+9x^2+y & 4y^3+x \\ 3x^2-1 & 2y+1 \end{pmatrix}$$

En (-2,2), $J_F = \begin{pmatrix} 6 & 30 \\ 11 & 5 \end{pmatrix}$ donc $J_F^{-1} = -\frac{1}{300}\begin{pmatrix} 5 & -30 \\ -11 & 6 \end{pmatrix}$ dont une valeur approchée est :

$$A = \begin{pmatrix} -0,02 & 0,1 \\ 0,03 & 0,02 \end{pmatrix}$$

D'où :

$$\phi(X) = \begin{pmatrix} x \\ y \end{pmatrix} - A \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$$, ce qui donne le programme :

```
10 X=-2 @ Y=2
20 PRINT X;Y
30 U=X*X
40 F=X*(Y+U*(3+X))+Y*Y*Y*Y
50 G=X*(U-1)+Y*(Y+1)
60 X=X+.02*F-.1*G
70 Y=Y-.03*F-.02*G
80 GOTO 20
```

Son exécution donne une stabilisation des chiffres pour :

$$x = -1{,}92105403697 \quad \text{et} \quad y = 1{,}82776656244$$

Il reste à voir quelle est la validité de ce résultat.

B. PRECISION DU RESULTAT

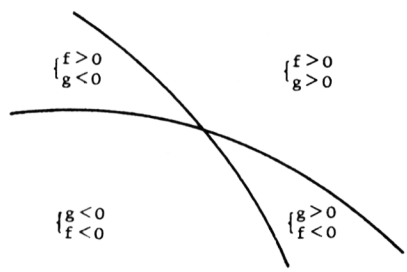
Il est possible de faire une étude directe de l'erreur en utilisant le théorème du point fixe (voir le chapitre 8) mais cela demande des calculs assez laborieux (voir cependant l'exercice 2).

Une vérification a posteriori, fondée sur le calcul de $F(\overline{X})$, est plus simple.

La courbe d'équation : $f(x,y) = 0$ partage le plan en deux domaines :

- l'ensemble des points (x,y) tels que $f(x,y) > 0$
- l'ensemble des points (x,y) tels que $f(x,y) < 0$.

Le calcul des signes de $f(\overline{x},\overline{y})$ et de $g(\overline{x},\overline{y})$ permet donc de situer le point $(\overline{x},\overline{y})$. La situation des points $(\overline{x},\overline{y})$, $(\overline{x}-\varepsilon,\overline{y})$, $(\overline{x},\overline{y}-\varepsilon)$, etc … , permet généralement de déterminer si la précision du calcul est bien ε.

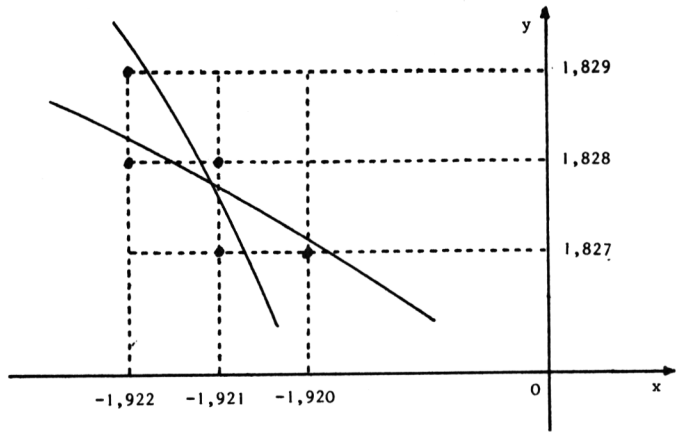


Ces calculs doivent donc être accompagnés de leurs calculs d'incertitude, ce que réalise le programme :

```
10 E=.0000000001
20 INPUT X,Y
30 U=X*X @ V=ABS(U)*E
40 Z=Y*Y @ A=Z*E
50 Z=Z*Y @ A=ABS(Y)*A+ABS(Z)*E
60 Z=Z*Y @ A=ABS(Y)*A+ABS(Z)*E
70 T=U*(3+X) @ B=ABS(3+X)*V+ABS
   (T)*E
80 T=Y+T @ B=B+ABS(T)*E
90 T=X*T @ B=ABS(X)*B+ABS(T)*E
100 F=T+Z @ A=A+B+ABS(F)*E
110 DISP F;A
120 Z=Y*(Y+1) @ A=ABS(Z)*E
130 T=X*(U-1) @ B=ABS(X)*V+ABS(T
    )*E
140 G=T+Z @ A=A+B+ABS(G)*E
150 DISP G;A
160 GOTO 20
```

où j'ai décomposé les calculs du programme précédent en opérations élémentaires accompagnées chacune de son calcul d'incertitude.

Ce qui donne avec cinq calculs :



Je peux donc affirmer, en utilisant la convexité des courbes, que $x = -1,921$ et $y = 1,828$ avec une incertitude de 10^{-3} (l'exercice 3 donne plus de détail sur le raisonnement sous-jacent).

De cette manière, il est possible d'affirmer que :

$$x = -1,921054 \quad \text{et} \quad y = 1,827767 \text{ à } 10^{-6} \text{ près.}$$

2. UN PROGRAMME POUR LES SYSTEMES 2×2.

A. PRINCIPE

En utilisant ce qui précède, il est possible de donner un programme général pour la résolution d'un système :

$$\begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases}$$

Toutefois, celui-ci ne calcule pas l'incertitude sur le résultat.

Le principe en est la méthode de Newton, c'est-à-dire que je définis une suite $(x_n,y_n)_{n\geq 0}$ par la donnée de (x_0,y_0) et la formule :

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ y_n \end{pmatrix} - J^{-1}(x_n,y_n) \begin{pmatrix} f(x_n,y_n) \\ g(x_n,y_n) \end{pmatrix}$$

où

$$J = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{pmatrix}$$ est la matrice jacobienne de : $\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} f(x,y) \\ g(x,y) \end{pmatrix}$.

D'où le programme :

B. LISTE DU PROGRAMME

```
 10 ! Resolution d'un systeme de
      2 equations a 2 inconnues
 20 !    ******************
 30 !
 40 ! Introduction des donnees:
 50 ! X,Y et la precision E
 60 INPUT X,Y,E
 70 !    Boucle de calcul
 80 !       ********
 90 F=X*X+Y*Y-1 ! f(x,y)
100 G=2*X+Y-1 ! g(x,y)
110 A=2*X ! df/dx
120 B=2*Y ! df/dy
130 C=2 ! dg/dx
140 D=1 ! dg/dy
150 ! Inversion de la matrice
          jacobienne J
160 Z=1/(A*D-B*C)
170 A=Z*A @ B=Z*B @ C=Z*C @ D=Z*
    D
180 ! Calcul des corrections z
          et t sur x et y:
190 !       (z)     -1 (f)
200 !       ( ) =-J    ( )
210 !       (t)        (g)
220 Z=-D*F+B*G @ T=C*F-A*G
230 !     Test d'arret
240 IF ABS(Z)+ABS(T)<E THEN 300
250 !   Calcul de x et y
260 X=X+Z @ Y=Y+T @ GOTO 70
270 !
280 !    Sortie du resultat
290 !      ***********
300 PRINT X;Y
310 END
```

Dans le programme, les fonctions f et g sont :

$$f(x,y) = x^2 + y^2 - 1 \quad \text{et} \quad g(x,y) = 2x + y - 1.$$

C. UTILISATION DU PROGRAMME

Il suffit d'introduire un point non trop éloigné du résultat et la précision désirée, ici :

1,0,1.E-6

Le résultat est : 0,8 et -0,6 qui est d'ailleurs exact.

Il est possible que le processus diverge, le programme précédent ne le signale pas mais il est facile de s'en douter à cause du temps d'exécution. Dans ce cas, il est nécessaire de choisir un autre point de départ.

Si vous voulez que votre ordinateur vous signale lui-même que le processus est divergent, il suffit de complèter le programme par :

```
35  K=100 : N=0
85  N=N+1
86  IF N>K THEN 320
320 PRINT "Processus divergent"
330 END
```

Enfin, pour utiliser le programme avec un autre système, n'oubliez pas de modifier les lignes 90 à 140.

3. SYSTEMES DE PLUS DE DEUX EQUATIONS

Ils se résolvent de la même façon, la matrice jacobienne pouvant alors être inversée par la méthode de Gauss (voir le chapitre 10). Pour limiter le temps de calcul, il y aura alors intérêt à faire un calcul unique comme au paragraphe 1.

4. EXERCICES

1. Résolvez les systèmes :

$$\begin{cases} x^2 + 4y^2 - 36 = 0 \\ x^2 + y^2 - 12x + 27 = 0 \end{cases} \qquad \begin{cases} x^3 - 3xy^2 - 2(x^2-y^2) - 11x + 52 = 0 \\ 3x^2y - y^3 - 4xy - 11y = 0 \end{cases}$$

$$\begin{cases} x + y + z = 6 \\ x^2 + y^2 + z^2 = 14 \\ x^3 + y^3 + z^3 = 36 \end{cases} \qquad \begin{cases} x + x^2 - 2yz = 0,1 \\ y - y^2 + 3xz = -0,2 \\ z + z^2 + 2xy = 0,3 \end{cases}$$

$$\begin{cases} \frac{\pi}{2} \frac{x(1-y^2z^2)(z^2-y^2)}{zy^2(1+z^2)} = 5 \\ 2x - \frac{10}{\pi} \operatorname{Log} \frac{1-z}{1+z} = 1 \\ x\left(y + \frac{1}{y}\right) - \frac{5}{\pi} \operatorname{Log} \frac{(1-yz)(z-y)}{(1+yz)(z+y)} = 2 \end{cases}$$

Dans chaque cas, vous déterminerez la précision du résultat.

2. Le contexte de cet exercice est le paragraphe 1.A. Soit $D = [-2;-1,5] \times [1,5;2]$, montrez que : $\phi(D) \subset D$. Calculez la matrice jacobienne J de ϕ puis montrez que :

$$\| J(x,y) \| \leqslant 0,5 \text{ pour tout } (x,y) \in D$$

(voir l'exercice 8 du chapitre 10 pour la définition de $\| J \|$) . Pour ces calculs, vous pourrez utiliser le chapitre 2 du tome 3.

En utilisant le théorème du point fixe, calculez le nombre d'itérations à effectuer pour calculer la racine à 10^{-3} près, comparez avec les résultats du paragraphe 1.B.

3. Le contexte de cet exercice est le paragraphe 1.B., $D = [-2;-1,5] \times [1,5;2]$ comme dans l'exercice 2.

Montrez que la relation : $f(x,y) = 0$ et $(x,y) \in D$ définit implicitement une fonction : $y = \varphi(x)$ sur $[-2;-1,5]$ à valeurs dans $[1,5;2]$ et une seule telle que : $f\left(x,\varphi(x)\right) = 0$ pour tout x. Montrez que φ est de classe C^1 et décroissante. De même associez une fonction ψ à g. Utilisez alors les calculs du paragraphe 1.B. pour conclure.

4. Programmez la résolution d'un système de n équations à n inconnues.

Index alphabétique

Masson Éditeur,
120, bd St-Germain
75280 Paris Cedex 06
Dépôt légal : septembre 1985

CORLET, IMPRIMEUR, S.A.
14110 Condé-sur-Noireau
N° d'Imprimeur : 6476
Dépôt légal : septembre 1985

Les derniers progrès technologiques ont mis les ordinateurs à la disposition de tous. Une nouvelle approche des Mathématiques est ainsi possible, les mettant à la portée du plus grand nombre.

La série « Les mathématiques par l'informatique individuelle » est née de cette constatation. Les auteurs y traitent de nombreux problèmes dont l'abord est facilité par l'outil informatique. L'approche pédagogique a été particulièrement soignée : chaque notion théorique et chaque programme sont mis à l'épreuve d'exemples concrets s'apparentant à des domaines divers : physique, chimie, mécanique, écologie, astronomie et cryptographie.

Après une initiation au langage Basic, ce premier tome traite de l'arithmétique et de ses applications à la cryptographie, de la résolution des équations et des systèmes d'équations ainsi que de leurs applications.

En particulier, il contient un programme de calcul des positions planétaires.

Tome 2. — Approximation, sommation, par D. JAKUBOWICZ et H. LEHNING.

Tome 3. — Optimisation, simulation, statistique, par D. JAKUBOWICZ et H. LEHNING.

Tome 4. — Graphisme, par H. LEHNING et D. JAKUBOWICZ.

9 782225 806971

ISBN : 2-225-80697-7

Document numérisé avec amour par

AMSTRAD CPC

MÉMOIRE ÉCRITE

https://acpc.me/