MASTER EN SCIENCES MATHÉMATIQUES

Erreurs d'arrondi dans les calculs numériques

Bertrand, Lucette; Deroo, Jacques

Award date:
1977

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES N.D. DE LA PAIX

N A M U R

Année académique 1976-1977

ERREURS D'ARRONDI

DANS LES CALCULS NUMERIQUES

Mémoire présenté pour l'obtention du grade
de Licencié en Sciences
mathématiques
par

BERTRAND Lucette
DEROO Jacques

Promoteur
J.P. THIRAN

Nous remercions vivement Monsieur J.P. THIRAN pour l'aide qu'il nous a apportée.

A V A N T - P R O P O S

Lorsqu'un problème comprenant des nombres réels est résolu par ordinateur, on ne peut pas déterminer avec certitude quel est le nombre de chiffres significatifs exacts du résultat obtenu. La raison en est simple. Les réels représentés en machine ne possèdent qu'un nombre donné de chiffres significatifs. Par conséquent, le produit, la somme et le quotient de deux de ces nombres n'auront pas, en général, une représentation machine exacte, ce qui entraîne une erreur appelée erreur d'arrondi. Cette erreur se propage au cours des opérations et le résultat final ne sera jamais qu'une approximation du résultat exact.

Si l'on ne peut pas connaître avec certitude la valeur de l'erreur d'arrondi, nous pourrons cependant en donner les bornes. Celles-ci sont plus ou moins grandes suivant, d'une part le nombre et le type d'opérations effectuées et, d'autre part la structure des nombres en machine et la façon dont les opérations fondamentales sont exécutées. Il est donc nécessaire, avant de procéder à une étude sur la précision des résultats, d'analyser les lois selon lesquelles un nombre réel qui n'est pas représentable en machine sera approximé par un nombre plutôt que par un autre, que ce nombre réel soit une donnée du problème ou le résultat d'une opération fondamentale.

Dans un premier chapitre, nous effectuerons cette analyse dans le cadre d'une approche axiomatique proposée par Kulisch [4-6]. Cette approche présente un grand intérêt, car elle nous permet de nous dégager d'un calculateur particulier. Cependant, comme les algorithmes du chapitre 2 sont programmés sur l'ordinateur Siemens 4004, nous particulariserons ensuite l'étude aux opérations en virgule flottante, en simple et en double longueur, de cet ordinateur.

Dans un second chapitre, nous appliquerons ces résultats au problème suivant . On considère un polynôme qui s'exprime sous la forme de deux polynômes factorisés :

$$p(x) = k_1 \prod_{i=1}^{n} (x - a_i) + k_2 \prod_{i=1}^{m} (x - b_i)$$

et on désire calculer ses racines. L'origine de ce problème provient notamment de la synthèse des filtres [10]. Nous verrons que la nature des résultats diffère énormément suivant la manière de procéder. L'analyse régressive [1] nous permet d'interpréter les résultats calculés comme la solution exacte du problème perturbé et la précision des résultats dépendra de la sensibilité des résultats aux variations des données et des variables intermédiaires et de l'importance de ces variations. Dans un algorithme donné, il sera important que les résultats soient moins sensibles aux variations des variables intermédiaires qu'aux variations des données : se pose alors le problème de l'algorithme adéquat. Nous illustrerons ceci par l'application de deux méthodes au problème de la factorisation du polynôme p(x). La méthode usuelle calculera les coefficients du polynôme et factorisera le polynôme. La méthode produit évitera le calcul des coefficients. Les fonctions algébriques [11] nous fourniront un moyen d'évaluer la sensibilité des racines aux variations des coefficients.

CHAP. I - LES OPERATIONS ARITHMETIQUES FONDAMENTALES SUR L'ORDINATEUR

1. INTRODUCTION.

Dans la littérature, nous avons rencontré plusieurs façons d'étudier l'approximation d'un nombre réel par un nombre flottant, la manière dont s'effectuent les opérations fondamentales entre flottants et les erreurs qui en découlent. Des résultats importants nous ont été donnés par Wilkinson [1] et Sterbenz [2] et nous nous sommes intéressés à une approche axiomatique faite principalement par Kulisch [3-7].

Notre étude des nombres flottants comporte trois parties. En premier lieu, nous verrons que l'approximation d'un nombre réel par un nombre flottant peut être considérée comme une application d'un ensemble dans un sous-ensemble. Ensuite, nous étudierons les opérations fondamentales effectuées par l'ordinateur et plus particulièrement par l'ordinateur Siemens 4004 [8]. Finalement, nous montrerons la non-validité des lois algébriques telles que l'associativité, la distributivité, ...

2. REPRESENTATION DES NOMBRES FLOTTANTS SUR ORDINATEUR.

2.1. Concept de grille.

Définition 1.

Soit M un ensemble non vide. La relation $\leq$ sur M est appelée relation d'ordre sur M, et nous dirons que $\{M, \leq\}$ est un ensemble ordonné, si et seulement si

1) $\forall x \in M \quad x \leq x$ (réflexive)

2) $\forall x,y,z \in M \quad x \leq y \text{ et } y \leq z \Longrightarrow x \leq z$ (transitive)

3) $\forall x,y \in M \quad x \leq y \text{ et } y \leq x \Longrightarrow x = y$ (antisymétrique)

Nous dirons que $\{M, \leq\}$ est un ensemble totalement ordonné si et seulement si

1) $\{M, \leq\}$ est un ensemble ordonné

2) $\forall x,y \in M \quad x \leq y \text{ ou } y \leq x$

Définition 2.

Soient $\{M, \leq\}$ un ensemble ordonné et un sous-ensemble S de M.

Pour tout $x \in M$, nous noterons

$$L_S(x) = \{y \in S \mid y \leq x\}$$

et $$U_S(x) = \{y \in S \mid y \geq x\}.$$

Le sous-ensemble S de M est appelé grille de M si et seulement si

1) $\forall x \in M, L_S(x) \neq \emptyset$ et $U_S(x) \neq \emptyset$

2) $\forall x \in M \quad \exists y \in L_S(x) \quad \forall z \in L_S(x) \quad z \leq y$

et $\exists y' \in U_S(x) \quad \forall z' \in U_S(x) \quad z' \geq y'$.

Propriété 1.

Les éléments y et y' sont uniques.

Preuve

Supposons que pour un x quelconque de M, y et y" sont des éléments de $L_S(x)$ qui vérifient

$$\forall z \in L_S(x) \quad z \leq y \text{ et } z \leq y".$$

En remplaçant z par y" et y, nous obtenons :

$$y" \leq y \text{ et } y \leq y".$$

Donc, par la propriété d'antisymétrie de la relation $\leq$,

$$y = y"$$

et nous avons vérifié l'unicité de y. L'unicité de y' se démontre de manière analogue.

Notations.

Les éléments y et y' seront notés respectivement $\max L_S(x)$ et $\min U_S(x)$.

2.2. Introduction aux nombres flottants.

Théorème 1. [9].

Soit un nombre entier $b > 1$.

Pour tout nombre réel y différent de zéro, il existe un entier n et une suite de nombres entiers $(c_i)_{i=1}^{\infty}$ tels que

$$y = * \sum_{i=1}^{\infty} c_i \, b^{n-i} = * \, c_1 \, c_2 \ldots c_n \cdot c_{n+1} \ldots$$

avec $* \in \{+,-\}$

$$1 \leq c_1 \leq b - 1$$

$$0 \leq c_i \leq b - 1 \qquad \text{pour } i > 1,$$

et il est impossible qu'il existe un entier k tel que, quel que soit l'entier $j > k$,

$$c_j = b - 1.$$

a) Version multiplicative du principe d'Archimède.

Si A et B sont deux nombres réels tels que $A > 1$ et $B > 0$, alors il existe un entier n et un seul vérifiant

$$A^{n-1} \leq B < A^n.$$

b) Démonstration du théorème.

Supposons $y > 0$. Alors b et y vérifient les hypothèses du principe d'Archimède. Par conséquent, il existe un entier n tel que $y \in [b^{n-1}, b^n[$.

Or $[b^{n-1}, b^n[= \bigcup_{i=1}^{b-1} [ib^{n-1}, (i+1) b^{n-1}[$

et les intervalles $[ib^{n-1}, (i+1) b^{n-1}[$ sont deux à deux disjoints. Il s'ensuit qu'il existe un et un seul entier c_1 tel que

$$y \in [c_1 b^{n-1}, (c_1 + 1) b^{n-1}[\quad \text{avec } 1 \leq c_1 \leq b - 1.$$

Or $[c_1 b^{n-1}, (c_1 + 1) b^{n-1}[=$

$$\bigcup_{i=0}^{b-1} [c_1 b^{n-1} + ib^{n-2}, c_1 b^{n-1} + (i+1) b^{n-2}[,$$

et les intervalles $\left[c_1\, b^{n-1} + i b^{n-2},\ c_1\, b^{n-1} + (i+1)\, b^{n-2}\right[$ sont deux à deux disjoints. Donc, il existe un et un seul entier c_2 tel que

$$y \in \left[c_1\, b^{n-1} + c_2\, b^{n-2},\ c_1\, b^{n-1} + (c_2 + 1)\, b^{n-2}\right[$$

avec $0 \leqslant c_2 \leqslant b - 1$.

En poursuivant indéfiniment le processus, nous obtenons une suite $(c_i)_{i=1}^{\infty}$ de nombres entiers tels que

$$1 \leqslant c_1 \leqslant b - 1$$

$$0 \leqslant c_i \leqslant b - 1 \qquad \text{pour } i > 1$$

et vérifiant

$$y = \sum_{i=1}^{\infty} c_i\, b^{n-i}.$$

Supposons à présent qu'il existe un entier k tel que, quel que soit l'entier $j > k$, $c_j = b - 1$.

Posons $A = \sum_{i=1}^{k} c_i\, b^{n-i}$. Par conséquent, y vérifie les inégalités suivantes :

$$A + (b - 1)\, b^{n-k-1} \leqslant y < A + b^{n-k}$$

$$A + (b - 1)\, b^{n-k-1} + (b - 1)\, b^{n-k-2} \leqslant y < A + b^{n-k}$$

$$\ldots\ldots\ldots\ldots\ldots$$

Donc :

$$y \in \bigcap_{i=1}^{\infty} \left[A + \sum_{s=1}^{i} (b - 1)\, b^{n-k-s},\ A + b^{n-k}\right[,$$

ce qui est impossible car

$$\bigcap_{i=1}^{\infty} \left[A + \sum_{s=1}^{i} (b - 1)\, b^{n-k-s},\ A + b^{n-k}\right[= \emptyset.$$

En conclusion, quel que soit le nombre entier k, on peut trouver un entier $j > k$ tel que $c_j \neq b - 1$. ∎

Considérons l'ensemble des nombres réels noté $\mathbb{R}$ et un nombre entier $b > 1$.

Définissons les ensembles S et M comme suit :

(D1) $S = \{ x \in \mathbb{R} \mid$ il existe $(c_i)_{i=1}^{p}$ et e entiers tels que $x = * \; 0 \, . \, c_1 \, c_2 \ldots\ldots\ldots\ldots c_p \, b^e$ noté $m \, b^e$ avec $* \in \{+,-\}$

$1 \leq c_1 \leq b - 1$

$0 \leq c_i \leq b - 1$ pour $i > 1$

$e_1 \leq e \leq e_2$

$e_1 \leq 0$ et $e_2 \geq 1$ entiers

ou $x = + \; 0 \, . \, 0 \, 0 \ldots\ldots\ldots\ldots 0 \, b^{e_1} \; (= 0) \}$,

(D2) et $M = \{ x \in \mathbb{R} \mid x = 0$ ou $B_1 \leq |x| \leq B_2$

avec $B_1 = 0 \, . \, 1 \, b^{e_1}$

$B_2 = 0 \, . \, \underbrace{(b-1)\,(b-1) \ldots\ldots (b-1)}_{p \text{ fois}} \, b^{e_2} \}$.

Nous désignerons les éléments de S par "nombres flottants normalisés" et nous appellerons

m : la mantisse ou partie fractionnaire de x

b : la base

e : l'exposant de x.

S est l'ensemble des nombres représentables en machine dont la mantisse, exprimée en base b, comporte au maximum p chiffres. L'exposant est borné inférieurement et supérieurement pour éviter un dépassement de la capacité de la machine.

Donc, S dépend de quatre constantes inhérentes à la machine et nous pouvons écrire :

$S = S \, (b, \, p, \, e_1, \, e_2)$.

Conséquences.

Considérons les ensembles S et M définis par (D1) et (D2). Ils vérifient les propriétés suivantes :

1) L'ensemble $\{M, \leq\}$ est totalement ordonné et S est un sous-ensemble de M.

2) Le sous-ensemble S est une grille de M.

Preuve

Vérifions que quel que soit $x \in M$, il existe $y \in L_S(x)$ et $y' \in U_S(x)$ tels que

$$\forall z \in L_S(x) \quad z \leq y \text{ et } \forall z' \in U_S(x) \quad z' \geq y' \qquad (1)$$

Si $x = 0$, alors $y = y' = 0$ vérifient (1).

Dans le cas où $x \neq 0$, par le théorème 1 et l'hypothèse $x \in M$, il existe $(c_i)_{i=1}^{\infty}$ et e entiers tels que

$$x = * 0 . c_1 c_2 \ldots b^e ; \quad * \in \{+,-\}, \quad e_1 \leq e \leq e_2,$$

$$1 \leq c_1 \leq b - 1 \quad \text{et} \quad 0 \leq c_i \leq b - 1 \text{ pour } i > 1.$$

Si, pour tout $i > p$, $c_i = 0$, alors $x \in S$ et $y = y' = x$ vérifient (1).

Par contre, s'il existe $i > p$ tel que $c_i \neq 0$, en supposant $x > 0$,

$$y = b^e \sum_{i=1}^{p} c_i b^{-i} \in L_S(x)$$

$$\text{et} \quad y' = b^e \left(\sum_{i=1}^{p} c_i b^{-i} + b^{-p}\right) \in U_S(x)$$

vérifient (1).

Dans le cas contraire, c'est-à-dire $x < 0$,

$$y = -b^e \left(\sum_{i=1}^{p} c_i b^{-i} + b^{-p}\right) \in L_S(x)$$

et $\quad y' = -b^e \sum_{i=1}^{p} c_i\, b^{-i} \in U_S(x)$

vérifient (1).

Donc, quel que soit $x \in M$, (1) est vérifié. (2)

De plus, tout élément $x \in M$ est tel que

$$-B_2 \in L_S(x) \quad \text{et} \quad B_2 \in U_S(x).$$

Par conséquent,

$$\forall x \in M \quad L_S(x) \neq \emptyset \text{ et } U_S(x) \neq \emptyset. \tag{3}$$

Il suit de (2) et (3) que S est une grille de M.

L'unicité de y et y' tels que nous venons de les définir est une conséquence de la propriété 1. Comme précédemment, nous les noterons respectivement max $L_S(x)$ et min $U_S(x)$. ■

3) La grille S de M est symétrique en ce sens que $\forall x \in S, -x \in S$.

Propriété 2.

Si T et S sont des grilles de M vérifiant $T \subset S \subset M$, alors T est une grille de S.

Preuve

Pour $x \in S$, ce qui implique $x \in M$, on a par hypothèse :

$$L_T(x) \neq \emptyset \text{ et } U_T(x) \neq \emptyset,$$

$$\exists y \in L_T(x) \quad \forall z \in L_T(x) \quad z \leq y$$

et $$\exists y' \in U_T(x) \quad \forall z' \in U_T(x) \quad z' \geq y'$$

Par conséquent, T est une grille de S. ∎

Considérons un entier $n \leq p$ et définissons les ensembles M, S et T de la façon suivante :

(F1) $T = T(b, n, e_1, e_2)$

(F2) $M = \{ x \in \mathbb{R} \mid x = 0 \text{ ou } B_1 \leq |x| \leq B_2$

avec $B_1 = 0.1\, b^{e_1}$

$B_2 = 0.\underbrace{(b-1)(b-1)\ldots(b-1)}_{n \text{ fois}}\, b^{e_2} \}$

(F3) et $S = S(b, p, e_1, e_2) \cap M.$

Les sous-ensembles S et T de M sont des grilles de M et $T \subset S \subset M$. Donc, l'ensemble des nombres flottants normalisés dont la mantisse comprend au maximum n chiffres en base b est une grille de l'ensemble des flottants dont la mantisse comporte au maximum p chiffres avec $p \geq n$.

2.3. Les "approximations".

Définition 3.

Soient $\{M, \leqslant\}$ un ensemble ordonné et $S \subset M$, S une grille de M.

L'application $\square : M \longrightarrow S$ est appelée "approximation" de M dans S si et seulement si

1) $\forall x, y \in M \quad x \leqslant y \Rightarrow \square(x) \leqslant \square(y)$ (monotone)

2) $\forall x \in S \quad \square(x) = x$ (optimale)

Nous appellerons $\square$: "approximation" par une valeur inférieure si, pour tout $x \in M$, $\square(x) \leqslant x$; par une valeur supérieure si, pour tout $x \in M$, $\square(x) \geqslant x$.

Théorème 2.

Soient $\{M, \leqslant\}$ un ensemble ordonné et $S \subset M$ une grille de M.

L'application $\square : M \longrightarrow S$ est une "approximation" par une valeur inférieure si et seulement si

$$\forall x \in M \quad \square(x) = \max L_S(x).$$

Par contre, $\square$ est une "approximation" par une valeur supérieure si et seulement si

$$\forall x \in M \quad \square(x) = \min U_S(x).$$

Preuve

Supposons que pour tout $x \in M$, $\square(x) = \max L_S(x)$. Si x et y sont des éléments de M tels que $x \leqslant y$, alors $L_S(x) \subset L_S(y)$. Donc, $\max L_S(x) \leqslant \max L_S(y)$.

Par conséquent, si $x \leqslant y$,

$$\square(x) \leqslant \square(y)$$

et la monotonicité de l'application $\square$ est vérifiée. (4)

Si x est un élément de S, alors $x \in L_S(x)$ et par hypothèse,

$$\forall y \in L_S(x) \quad x \geqslant y.$$

Donc $x = \max L_S(x)$ et, par conséquent, si $x \in S$

$$x = \square(x)$$

et l'optimalité de $\square$ est vérifiée. (5)

Par définition

$$\forall y \in L_S(x) \quad y \leqslant x.$$

Donc $x \geqslant \max L_S(x)$ ou de manière équivalente,

$$x \geqslant \square(x). \tag{6}$$

Il suit de (4), (5) et (6) que $\square$ est une "approximation" par une valeur inférieure.

Inversément, si $\square$ est une "approximation" par une valeur inférieure,

$$\square(x) \leqslant x$$

où $\square(x)$ est un élément de $L_S(x)$.

Supposons qu'il existe $y \in L_S(x)$ tel que

$$\square(x) \leqslant y \leqslant x.$$

Par la monotonicité de l'application $\square$,

$$\square(\square(x)) \leqslant \square(y) \leqslant \square(x).$$

Or $\square$ est optimal et $\square(x)$, y sont des éléments de $L_S(x)$. Donc

$$\square(x) \leqslant y \leqslant \square(x).$$

Ou encore $\square(x) = y$. Et finalement

$$\square(x) = \max L_S(x).$$

L'autre partie du théorème se démontre de manière analogue. ∎

Corollaire.

Il existe une et une seule "approximation" par une valeur inférieure et une seule "approximation" par une valeur supérieure d'un ensemble ordonné $\{M, \leq\}$ dans une grille S de M.

Notations.

Nous noterons $\max L_S(x)$ et $\min U_S(x)$ respectivement $\nabla(x)$ et $\Delta(x)$.

Soient S et M définis par (D1) et (D2). Pour tout $x \in M$,

$$\nabla(x) = -\Delta(-x)$$

et $$\Delta(x) = -\nabla(-x).$$

Preuve

Suivant les notations employées précédemment et les conséquences tirées des définitions de S et M par (D1) et (D2),

$$\nabla(x) = \max L_S(x)$$

Or, si $x = b^e \sum_{i=1}^{\infty} c_i\, b^{-i}$,

$$\max L_S(x) = b^e \sum_{i=1}^{p} c_i\, b^{-i} = -\min U_S(-x).$$

Par contre, si $x = -b^e \sum_{i=1}^{\infty} c_i\, b^{-i}$,

$$\max L_S(x) = -b^e \left(\sum_{i=1}^{p} c_i\, b^{-i} + b^{-p}\right) = -\min U_S(-x).$$

Donc $\nabla(x) = -\Delta(-x)$.

La seconde égalité se démontre de la même manière. ∎

Théorème 3.

Soient $\{M,\leqslant\}$ un ensemble ordonné, S et T des grilles de M telles que $T \subset S \subset M$.

Si $\nabla, \nabla_1, \nabla_2$ sont les "approximations" par une valeur inférieure qui vérifient :

$$\nabla_1 : M \longrightarrow S$$
$$\nabla_2 : S \longrightarrow T$$
$$\nabla : M \longrightarrow T$$

alors, pour tout $x \in M$,

$$\nabla(x) = \nabla_2(\nabla_1(x)).$$

Preuve

Etant donné que $T \subset S$, pour tout $x \in M$,

$$L_T(x) \subset L_S(x).$$

Il s'ensuit :

$$\nabla(x) = \max L_T(x) \leqslant \nabla_1(x) = \max L_S(x).$$

L'application ∇_2 est monotone et optimale, donc

$$\nabla_2(\nabla(x)) = \nabla(x) \leqslant \nabla_2(\nabla_1(x)). \qquad (7)$$

Par hypothèse, ∇_1 et ∇_2 sont des "approximations" par une valeur inférieure, ce qui entraîne

$$\nabla_2(\nabla_1(x)) \leqslant \nabla_1(x) \leqslant x.$$

L'application ∇ est monotone et optimale. Donc

$$\nabla(\nabla_2(\nabla_1(x))) = \nabla_2(\nabla_1(x)) \leqslant \nabla(x). \qquad (8)$$

Il suit de (7) et (8)

$$\nabla_2(\nabla_1(x)) = \nabla(x).$$

Remarque.

Le théorème se vérifie aisément dans le cas d'"approximations" par une valeur supérieure.

Soient T, M et S définis par (F1), (F2) et (F3). Les sous-ensembles S et T de M sont des grilles de M telles que $T \subset S \subset M$.

Nous pouvons donc appliquer le théorème à ces ensembles particuliers.

Si $\nabla, \nabla_1, \nabla_2$ "approximent" un nombre réel par un nombre flottant de valeur inférieure et

$$\nabla : M \longrightarrow T$$

$$\nabla_1 : M \longrightarrow S$$

$$\nabla_2 : S \longrightarrow T$$

alors, pour tout réel $x \in M$,

$$\nabla(x) = \nabla_2(\nabla_1(x)).$$

Si $\Delta, \Delta_1, \Delta_2$ "approximent" un nombre réel par un nombre flottant de valeur supérieure et

$$\Delta : M \longrightarrow T$$

$$\Delta_1 : M \longrightarrow S$$

$$\Delta_2 : S \longrightarrow T$$

alors, pour tout réel $x \in M$,

$$\Delta(x) = \Delta_2(\Delta_1(x)).$$

2.4. Les "approximations" dans les ensembles totalement ordonnés.

Théorème 4.

Soit $\{M,\leq\}$ un ensemble totalement ordonné. Si le sous-ensemble S de M est une grille de M et si les applications $\nabla : M \longrightarrow S$ et $\Delta : M \longrightarrow S$ sont les "approximations" respectivement par une valeur inférieure et par une valeur supérieure, alors, quel que soit $x \in M$, il n'existe pas d'élément $y \in S$ vérifiant

$$\nabla(x) < y < \Delta(x).$$

Preuve

Supposons qu'un tel y existe. Or $y \in S$ implique $y \in M$ et M est un ensemble totalement ordonné. Donc

$$\forall x \in M \quad y \leq x \quad \text{ou} \quad y \geq x.$$

Supposons $y \leq x$. Cela entraîne

$$\nabla(x) < y \leq x.$$

En raison des caractères monotone et optimal de l'application ∇

$$\nabla(x) < y \leq \nabla(x),$$

ce qui est impossible. Donc, il n'existe pas d'élément $y \in S$ vérifiant $\nabla(x) < y < \Delta(x)$. ∎

Théorème 5.

Soient $\{M,\leq\}$ un ensemble totalement ordonné, une grille S de M, une "approximation" $\square : M \longrightarrow S$ et les "approximations" de M dans S par une valeur inférieure et par une valeur supérieure notées respectivement ∇ et Δ.

Pour tout $x \in M$, considérons $I = \{y \in M \mid \nabla(x) \leq y \leq \Delta(x)\}$.

Si $x \in S$, alors $\square(x) = \nabla(x) = \Delta(x) = x$ et si $x \notin S$, alors on peut construire I_1 et I_2, des sous-ensembles de I

tels que

1) $I_1 \cup I_2 = I$

2) $\forall x_1 \in I_1 \quad \forall x_2 \in I_2 \quad x_1 < x_2$

3) si $x \in I_1 \quad \square(x) = \nabla(x)$

4) si $x \in I_2 \quad \square(x) = \Delta(x)$

Preuve

Considérons un élément x de M. Si $x \in S$, alors, en raison de l'optimalité des applications $\square$, ∇ et Δ,

$$\square(x) = \nabla(x) = \Delta(x) = x.$$

Prenons le cas où $x \notin S$. Par définition des applications ∇ et Δ

$$\nabla(x) \leq x \leq \Delta(x).$$

Or $\square$ est monotone et optimal. Donc

$$\square(\nabla(x)) = \nabla(x) \leq \square(x) \leq \Delta(x) = \square(\Delta(x)).$$

Et, par le théorème 4,

$$\square(x) = \nabla(x) \quad \text{ou} \quad \square(x) = \Delta(x).$$

Définissons I_1 et I_2 de la façon suivante :

$$I_1 = \{y \in I \mid \square(y) = \nabla(x)\}$$

$$I_2 = \{y \in I \mid \square(y) = \Delta(x)\}.$$

De ces définitions, il découle immédiatement

$$I = I_1 \cup I_2$$

si $x \in I_1 \quad \square(x) = \nabla(x)$, mais

si $x \in I_2 \quad \square(x) = \Delta(x)$.

Il nous reste à prouver que, quels que soient $x_1 \in I_1$ et $x_2 \in I_2$, $x_1 < x_2$.

Par l'absurde, supposons qu'il existe $x_1 \in I_1$ et $x_2 \in I_2$ tels que $x_1 \gtrdot x_2$.

Or, x_1 et x_2 sont des éléments de I, donc

$$\nabla(x) \lessdot x_2 \lessdot x_1 \lessdot \Delta(x).$$

En appliquant l'"approximation" $\square$ à ces inégalités, nous obtenons :

$$\square(\nabla(x)) = \nabla(x) \lessdot \square(x_2) \lessdot \square(x_1) \lessdot \Delta(x) = \square(\Delta(x)).$$

Par hypothèse, $x_1 \in I_1$ et $x_2 \in I_2$, donc

$$\nabla(x) \lessdot \Delta(x) \lessdot \nabla(x) \lessdot \Delta(x).$$

ou $\nabla(x) = \Delta(x)$.

Ceci est impossible étant donné que $x \notin S$ et la démonstration du théorème est achevée. ∎

2.5. Les "approximations" symétriques.

Définition 4.

Soient M une partie de $\mathbb{R}$, S une grille symétrique de M et $\square : M \longrightarrow S$ une "approximation" de M dans S.

L'"approximation" $\square$ sera dite symétrique si, par définition,

$$\forall x \in M \qquad \square(-x) = -\square(x).$$

Soient S et M définis par (D1) et (D2). Nous allons reprendre les notations du théorème 5 pour définir différentes "approximations" symétriques utilisées sur ordinateur.

1) arrondi : R(x).

Si $x \geqslant 0$

$$I_1 = \left[\nabla(x), \frac{\nabla(x) + \Delta(x)}{2}\right[$$

$$I_2 = \left[\frac{\nabla(x) + \Delta(x)}{2}, \Delta(x)\right].$$

Si $x < 0$

$$I_1 = \left[\nabla(x), \frac{\nabla(x) + \Delta(x)}{2}\right]$$

$$I_2 = \left]\frac{\nabla(x) + \Delta(x)}{2}, \Delta(x)\right].$$

2) Approximation éloignée de zéro : $\Upsilon(x)$.

Si $x \geqslant 0$

$$I_1 = \{\nabla(x)\}$$

$$I_2 = \left]\nabla(x), \Delta(x)\right]$$

Si $x < 0$

$$I_1 = \left[\nabla(x), \Delta(x)\right[$$

$$I_2 = \{\Delta(x)\}.$$

3) Approximation dirigée vers zéro : $\varphi(x)$.

Si $x \geqslant 0$

$$I_1 = \left[\nabla(x), \Delta(x)\right[$$

$$I_2 = \{\Delta(x)\}.$$

Si $x < 0$

$$I_1 = \{\nabla(x)\}$$

$$I_2 = \left]\nabla(x), \Delta(x)\right].$$

2.6. Etude de l'erreur.

1) Concept d'erreur relative.

Soit $x \neq 0$ et $\bar{x}$ une approximation de x. On définit l'erreur relative ε par

$$\varepsilon = \frac{\bar{x} - x}{x}$$

ou, de façon équivalente,

$$\bar{x} = x(1 + \varepsilon).$$

2) Bornes d'erreurs dues aux "approximations" ∇ et Δ.

Soient S et M définis par (D1) et (D2) et $x \in M$. Si $x \in S$, alors $\nabla(x) = \Delta(x)$ et l'erreur relative ε est égale à zéro.

Supposons alors que $x \notin S$. Donc, il existe $(c_i)_{i=1}^{\infty}$ et e entiers tels que

$$x = * \, b^e \sum_{i=1}^{\infty} c_1 \, b^{-i}, \quad * \in \{+,-\}.$$

Si $\bar{x}$, l'approximation de x, est définie par

$$\bar{x} = * \, b^e \sum_{i=1}^{p} c_i \, b^{-i}, \quad * \in \{+,-\}$$

alors, l'erreur relative ε vaut :

$$\varepsilon = \frac{-\sum_{i=p+1}^{\infty} c_i \, b^{-i}}{\sum_{i=1}^{\infty} c_i \, b^{-i}}.$$

Or $\sum_{i=p+1}^{\infty} c_i \, b^{-i} < b^{-p}$ et $\sum_{i=1}^{\infty} c_i \, b^{-i} \geq b^{-1}$.

Nous pouvons donc en déduire

$$|\varepsilon| < b^{1-p}.$$

Si, par contre, $\bar{x}$ est défini par

$$\bar{x} = * \, b^e \, (\sum_{i=1}^{p} c_i \, b^{-i} + b^{-p}),$$

alors, l'erreur relative ε vaut

$$\varepsilon = \frac{- \sum_{i=p+1}^{\infty} c_i \, b^{-i} + b^{-p}}{\sum_{i=1}^{\infty} c_i \, b^{-i}} .$$

Or, $0 < \sum_{i=p+1}^{\infty} c_i \, b^{-i} < b^{-p}$ puisque $x \notin S$.

Ce qui entraîne

$$0 < - \sum_{i=p+1}^{\infty} c_i \, b^{-i} + b^{-p} < b^{-p}.$$

De plus,

$$\sum_{i=1}^{\infty} c_i \, b^{-i} \geqslant b^{-1}.$$

Par conséquent, $|\varepsilon| < b^{1-p}$.

3) Bornes d'erreurs dues aux "approximations" R, Υ et Φ.

En utilisant les bornes d'erreurs obtenues pour les "approximations" ∇ et Δ, nous obtenons les résultats suivants :

a) $\left|\frac{R(x) - x}{x}\right| < \frac{1}{2} \, b^{1-p}$.

b) $\left|\frac{\Psi(x) - x}{x}\right| < b^{1-p}$.

c) $\left|\frac{\Phi(x) - x}{x}\right| < b^{1-p}$.

2.7. Représentation des nombres flottants sur l'ordinateur Siemens 4004.

L'ordinateur Siemens 4004 travaille en base 16. La conversion des nombres décimaux en nombres hexadécimaux s'effectue au moyen du système de conversion de base représenté ci-dessous :

Système de conversion de base.

Décimal base 10	Hexadécimal base 16	Décimal base 10	Hexadécimal base 16
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

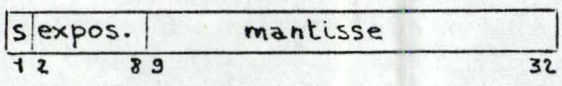
Deux sortes de nombres flottants sont pris en considération : les flottants de longueur simple et ceux de longueur double. Les premiers comportent 32 bits (ou huit chiffres hexadécimaux) et les seconds 64 bits (ou seize chiffres hexadécimaux) qui se répartissent comme suit :

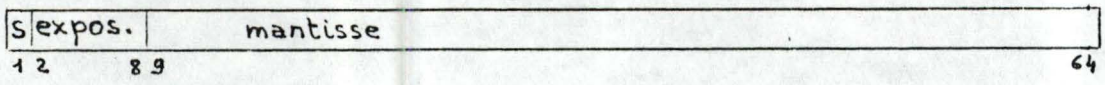
1 bit pour le signe de la mantisse

7 bits pour l'exposant

24 bits pour la mantisse en simple longueur

56 bits pour la mantisse en double longueur.





Le point de la fraction est supposé se trouver à gauche du premier chiffre de la mantisse.

Etant donné que 7 bits servent à représenter l'exposant, il existe au maximum 128 représentations possibles. Pour ne pas l'affecter d'un signe, l'exposant est incrémenté d'une valeur constante égale à 64 en base 10 (ou 40 en base 16).

Ce nombre, exposant $+ 64_{10}$, est appelé caractéristique.

CARACTERISTIQUE		EXPOSANT
Décimal	Hexadécimal	Décimal
0	0	-64
64	40	0
127	7F	+63

Les nombres flottants normalisés représentables sur l'ordinateur Siemens sont donc des éléments de S = S(16, 6, -64, 63) s'ils sont en simple longueur, de S = S(16, 14, -64, 63) s'ils sont en double longueur.

La représentation flottante normalisée du nombre zéro sera donnée par le signe +, une caractéristique et une mantisse nulles.

L'ordinateur Siemens utilise l'"approximation" dirigée vers zéro. Par conséquent, si x est une donnée d'un problème, sa représentation en machine est $\varphi(x)$.

3. L'ARITHMETIQUE SUR ORDINATEUR.

3.1. Point de vue axiomatique.

Définition 5.

Soit un ensemble M non vide. Définissons sur M une loi interne notée $*$: $M \times M \longrightarrow M$. Nous appellerons l'ensemble M groupoïde et nous le noterons $\{M,*\}$.

Définition 6.

Soit un groupoïde $\{M,*\}$. L'ensemble M possède un neutre à droite pour la loi $*$ s'il existe un élément $e \in M$ tel que

$$\forall x \in M \quad x * e = x.$$

L'ensemble M possède un neutre à gauche pour la loi $*$ s'il existe un élément $e \in M$ tel que

$$\forall x \in M \quad e * x = x.$$

Si e est à la fois neutre à gauche et à droite, nous l'appellerons neutre de M pour la loi $*$.

Définition 7.

Soient $\{M,*\}$ un groupoïde, $\{M,\leq\}$ un ensemble ordonné et une grille S de M.

Définissons sur S une loi interne $\boxtimes$: $S \times S \longrightarrow S$. Cette loi donne à $\{S,\boxtimes\}$ une structure de groupoïde. Sous ces hypothèses, introduisons les définitions suivantes :

1) $\{S,\boxtimes\}$ respecte la structure d'ordre sur M par rapport à la loi $*$ si

$$\forall x,y,z,t \in S \quad x * y \leq z * t \Longrightarrow x \boxtimes y \leq z \boxtimes t.$$

2) $\{S,\boxtimes\}$ est un groupoïde optimal si

$$\forall x,y \in S \quad x * y \in S \Longrightarrow x \boxtimes y = x * y.$$

3) Dans le cas particulier où M est une partie de $\mathbb{R}$, $\{S, \boxtimes\}$ est un groupoïde dirigé vers zéro si, pour tout $x,y \in S$,

$$x * y < 0 \Longrightarrow x \boxtimes y \geq x * y$$

$$x * y \geq 0 \Longrightarrow x \boxtimes y \leq x * y.$$

Théorème 6.

Soient $\{M, *\}$ un groupoïde, $\{M, \leq\}$ un ensemble ordonné, une grille S de M et $\square_* : M \longrightarrow S$ une application de M dans S. Définissons une loi interne $\boxtimes : S \times S \longrightarrow S$ de la façon suivante :

$$\forall x,y \in S \quad x \boxtimes y = \square_*(x * y).$$

1) Si $\square_*$ est une application monotone, alors $\{S, \boxtimes\}$ respecte la relation d'ordre sur M par rapport à la loi $*$.

2) Si $\square_*$ est une application optimale, alors $\{S, \boxtimes\}$ est un groupoïde optimal.

3) Si la loi $*$ est commutative dans M, alors la loi $\boxtimes$ est commutative dans S.

4) Si $M \subset \mathbb{R}$ et, pour tout $x \in M$

$$x < 0 \Longrightarrow \square_*(x) \geq x$$

$$x \geq 0 \Longrightarrow \square_*(x) \leq x,$$

alors $\{S, \boxtimes\}$ est un groupoïde dirigé vers zéro.

Preuve

1) Soient $x,y,z,t \in S$ tels que $x * y \leq z * t$. Les éléments $x * y$ et $z * t$ sont des éléments de M et, par hypothèse, $\square_*$ est monotone. Donc

$$\square_*(x * y) \leq \square_*(z * t).$$

Il suit de la définition de la loi ⊠ :

$$x \boxtimes y \leq z \boxtimes t.$$

Par conséquent, $\{S, \boxtimes\}$ respecte la relation d'ordre sur M par rapport à la loi $*$.

2) Soient $x,y \in S$ tels que $x * y \in S$. L'application $\square_*$ est optimale, donc

$$\square_*(x * y) = x * y.$$

Il s'ensuit

$$x \boxtimes y = x * y.$$

Dès lors, $\{S, \boxtimes\}$ est un groupoïde optimal.

3) Soient $x,y \in S$. Par hypothèse,

$$x \boxtimes y = \square_*(x * y) \quad \text{et} \quad y \boxtimes x = \square_*(y * x).$$

Or la loi $*$ est commutative dans M. Donc

$$\square_*(x * y) = \square_*(y * x).$$

Par conséquent, la loi ⊠ est commutative dans S.

4) Considérons les ensembles S et M tels que $S \subset M \subset \mathbb{R}$ et les éléments $x,y \in S$.

L'élément $x * y \in M$. Si $x * y < 0$, alors, par hypothèse,

$$x \boxtimes y = \square_*(x * y) \geq x * y.$$

Par contre, si $x * y \geq 0$,

$$x \boxtimes y = \square_*(x * y) \leq x * y.$$

Donc, $\{S, \boxtimes\}$ est un groupoïde dirigé vers zéro. ∎

Théorème 7.

Soient $\{M, *\}$ un groupoïde qui possède un neutre à droite noté e et $S \subset M$, une grille de M telle que $e \in S$. Si la loi $\boxtimes : S \times S \longrightarrow S$ donne à $\{S, \boxtimes\}$ une structure de groupoïde optimal, alors e est un neutre à droite de S pour la loi $\boxtimes$.

Preuve

Soit un élément $x \in S \subset M$. L'élément e est neutre à droite de M, donc

$$x * e = x.$$

Or, par hypothèse, $\{S, \boxtimes\}$ est un groupoïde optimal et $x \in S$. Dès lors,

$$x \boxtimes e = x * e = x.$$

Par conséquent, e est neutre à droite de S pour la loi $\boxtimes$. ∎

3.2. Etude des opérations arithmétiques sur l'ordinateur Siemens 4004.

Considérons les ensembles $S = S(b,p,e_1,e_2)$ et M définis par (D1) et (D2) (voir section 2.2.), Υ et φ les "approximations" respectivement éloignée de zéro et dirigée vers zéro. Introduisons un nouvel ensemble noté T et défini par

$$T = T(b,p-1,e_1,e_2).$$

Cet ensemble nous sera utile par la suite. Nous noterons Υ^* et φ^* les "approximations" de M dans T respectivement éloignée de zéro et dirigée vers zéro. Les opérations fondamentales +,-,. et / telles qu'elles sont exécutées par l'ordinateur seront notées $\boxplus$, $\boxminus$, $\boxdot$ et $\boxslash$.

Le problème est le suivant : soient $* \in \{+,-,.,/\}$ et $\circledast \in \{\boxplus, \boxminus, \boxdot, \boxslash\}$, quelle est l'application $\square_*$ qui vérifie

$$\forall x,y \in S \quad (S \setminus \{0\} \text{ si } * = /) \quad x \circledast y = \square_*(x * y) \quad ?$$

De la connaissance des applications $\square_*$, nous déduirons des propriétés relatives aux lois $\boxplus$, $\boxminus$, $\boxdot$, $\boxslash$ et nous trouverons une borne de l'erreur due à l'approximation de $x * y$ par $x \circledast y$.

Pour plus de facilités, nous ne considérerons pas, au cours de cette étude, le problème de dépassement de capacité. Quel que soit $x \circledast y = mb^e$, nous supposerons $e_1 \leq e \leq e_2$. Le dépassement de ces bornes provoque l'arrêt de l'exécution d'un programme. Les termes "underflow" et "overflow" désignent respectivement un dépassement vers le bas ($e < e_1$) et un dépassement vers le haut ($e > e_2$).

Et enfin, nous dirons qu'un nombre flottant est "shifté" vers la droite (ou vers la gauche) d'un chiffre hexadécimal quand il subit les transformations suivantes :

1) la mantisse est déplacée vers la droite (ou vers la gauche) d'un chiffre hexadécimal.

2) le premier chiffre de la mantisse devient zéro (le dernier dans le cas d'un "shifting" vers la gauche).

3) la caractéristique est augmentée (ou diminuée) d'une unité.

Un nombre flottant $mb^e \neq 0$ est normalisé par un "shifting" vers la gauche ou vers la droite lorsque

$$|m| < b^{-1} \quad \text{ou} \quad |m| \geq 1.$$

3.3. Addition et soustraction en virgule flottante.

3.3.1. Description des opérations sur l'ordinateur Siemens.

La soustraction sur ordinateur consiste en une addition après changement de signe du nombre à soustraire. Dès lors

$$x \boxminus y = x \boxplus (-y).$$

C'est la raison pour laquelle nous nous limiterons à l'étude du processus de l'addition.

1) Addition de deux nombres flottants de longueur simple.

Soient $x, y \in S(16,6,-64,63)$.

lère étape.

Les caractéristiques des deux nombres sont comparées l'une à l'autre. Si elles ne sont pas identiques, le nombre ayant la caractéristique la plus petite est "shifté" vers la droite jusqu'à ce que les exposants soient égaux. Seuls les sept premiers chiffres de la mantisse de ce nombre sont conservés.

2ème étape.

Les mantisses sont additionnées pour former la mantisse de la somme intermédiaire. Celle-ci consiste en un signe et sept chiffres hexadécimaux plus, éventuellement, un report (elle peut être, en valeur absolue, supérieure à 1). Le septième chiffre est appelé chiffre de garde. Il conserve une partie intéressante de l'addition exacte quand il y a eu "shifting" avant l'addition.

3ème étape.

La somme intermédiaire est ensuite normalisée et seuls sont gardés les six premiers chiffres de la mantisse.

Exemples.

a) $0.FFF987\ 16^{0} \boxplus 0.F83768\ 16^{-2} = 0.100F1B\ 16^{1}$

En effet, la somme intermédiaire est
$0.FFF987\ 16^{0} + 0.00F8376\ 16^{0} = 1.00F1BE6\ 16^{0}$.

Ensuite, le résultat intermédiaire est normalisé et sa mantisse est tronquée à six chiffres.

b) $0.105368\ 16^{0} \boxminus 0.673457\ 16^{-2} = 0.FEC33B\ 16^{-1}$
s'obtient de la manière suivante : le résultat intermédiaire est
$0.105368\ 16^{0} - 0.0067345\ 16^{0} = 0.0FEC33B\ 16^{0}$;
le résultat final est obtenu après normalisation du résultat intermédiaire.

Remarque : Le chiffre de garde ne se conserve pas au cours d'une série d'additions.

2) Addition de deux nombres flottants de longueur double.

Soient deux éléments de S(16,14,-64,63).

1ère étape.

Si les caractéristiques ne sont pas égales, elles le deviennent après un shifting à droite du nombre ayant la caractéristique la plus petite. Seuls les quatorze premiers chiffres de la mantisse sont conservés.

2ème étape.

Les mantisses sont additionnées pour former la mantisse de la somme intermédiaire. Celle-ci consiste en un signe et quatorze chiffres hexadécimaux plus, éventuellement, un report. Il n'y a donc pas de chiffre de garde.

3ème étape.

La somme intermédiaire est normalisée et sa mantisse est tronquée à quatorze chiffres hexadécimaux.

Exemples.

a) $0.1\ 16^{0} \boxminus 0.FFFFFFFFFFFFF\ 16^{-1} = 0.1\ 16^{-13}$

En effet, le résultat intermédiaire est
$0.1\ 16^{0} - 0.0FFFFFFFFFFFF\ 16^{0} =$
$0.000000000000001\ 16^{0}$.

Le résultat final est obtenu après normalisation du résultat intermédiaire.

Remarque : le résultat exact de la soustraction est $0.1\ 16^{-14}$. S'il y avait eu un chiffre de garde, c'est ce résultat que nous aurions obtenu.

b) $0.FF4\ 16^{0} \boxplus 0.196314ABCD1378\ 16^{-1} =$
$0.100D6314ABCD13\ 16^{1}$.

En effet, la somme intermédiaire est
$0.FF4\ 16^{0} + 0.0196314ABCD137\ 16^{0} =$
$1.00D6314ABCD137\ 16^{0}$.

Le résultat intermédiaire est normalisé et seuls les quatorze premiers chiffres de la mantisse sont gardés.

Remarque : l'existence d'un chiffre de garde n'aurait pas accru la précision de ce résultat.

3.3.2. Etude du processus de l'addition.

Pour pouvoir étudier simultanément le processus de l'addition en simple et en double longueur, nous introduisons le nombre q. Celui-ci vaut 1 lorsque l'addition s'effectue avec l'aide d'un chiffre de garde et 0 dans le cas contraire.

Soient x et y deux nombres flottants signés

$$x = mb^e \quad \text{et} \quad y = nb^f$$

où m,n et e,f sont respectivement les mantisses et exposants de x et y. Nous supposerons $|x| \geq |y|$. Si tel n'est pas le cas, nous nous y ramenons en échangeant les rôles de x et y.

Il est possible que x et y soient nuls et alors

$$x = +0.00\ldots b^{e_1} = y.$$

Par conséquent, $x \boxplus y = +0.00\ldots b^{e_1}$.

Considérons $x \neq 0$. Désignons par n' la mantisse obtenue après le déplacement de n de e-f chiffres, c'est-à-dire

$$n' = b^{-(e-f)} n,$$

et par n" la mantisse qui consiste en le signe et les p+q premiers chiffres de n'.

La mantisse u' de la somme intermédiaire est

$$u' = m + n",$$

et u est la mantisse obtenue après normalisation de u'. Le nombre entier g sera défini par

$$x \boxplus y = \varphi(ub^g).$$

Nous allons séparer notre étude en deux cas.

1) Les nombres x et y sont de même signe.

Dans ce cas, $|u'| = |m+n"| < 2$ et $|u'| \geq |m| \geq b^{-1}$. Si $b^{-1} \leq |u'| < 1$ alors $u' = u$. Il s'ensuit

$$x \boxplus y = \varphi(ub^e).$$

Par contre, si $|u'| \geq 1$ alors $u = u'b^{-1}$. Donc

$$x \boxplus y = \varphi(ub^{e+1}).$$

Dans les deux cas, les p premiers chiffres de u sont aussi les p premiers chiffres de m+n'. Par conséquent,

$$x \boxplus y = \Upsilon(x + y).$$

2) Les nombres x et y sont de signe contraire.

Dans ce cas, $|u'| = |m+n''| \leq |m| < 1$.

Supposons $u' \neq 0$. Ce sera toujours le cas si $x \neq y$. En effet, $u' = 0$ quand $m+n' = 0$. Or $b^{-1} \leq |m| < 1$, donc $b^{-1} \leq |n'| < 1$. Par conséquent, y n'a pas subi de "shifting" avant d'être additionné à x, c'est-à-dire $e = f$ et $n'' = n$. Puisque $m + n = 0$, $x = -y$.

Si $u' \neq 0$, plusieurs cas sont possibles.

a) $e - f \leq q$.

Alors $n'' = n'$ et $u' = m + n'$.
Par conséquent, $x \boxplus y = \Upsilon(x + y)$.

b) $q = 1$ et $e - f > 1$.

Alors m et n" vérifient les inégalités suivantes :

$$b^{-1} \leq |m| < 1 \quad \text{et} \quad 0 \leq |n''| < b^{-2}.$$

Donc $b^{-2} < |m| - |n''| < 1$. On en déduit $b^{-2} < |u'| < 1$.

Si $ub^g \in S$, alors $x \boxplus y = \Upsilon(x + y)$. Par contre, si $ub^g \notin S$, alors $x \boxplus y = \Upsilon(x + y)$.

Démontrons la première affirmation.

Supposons $b^{-1} \leq |u'| < 1$. Alors $u' = u$. De plus, n" et n' vérifient

$$|n''| \leq |n'| < |n''| + b^{-p}.$$

Donc $|m| - |n''| - b^{-p} < |m| - |n'| \leq |m| - |n''|$.

Par conséquent,

$$|ub^g| - b^{-(p-g)} < |x + y| \leq |ub^g| \text{ avec } g = e.$$

Par contre, si $b^{-2} < |u'| < b^{-1}$, alors $u'b = u$.

Les mantisses n" et n' vérifient

$$|n"| \leq |n'| < |n"| + b^{-(p+1)}.$$

Par conséquent,

$$|ub^g| - b^{-(p-g)} < |x + y| \leq |ub^g| \text{ avec } g = e-1.$$

Etant donné que $ub^g \in S$,

$$x \boxplus y = \Psi(ub^g) = ub^g.$$

Donc, $|x \boxplus y| - b^{-(p-g)} < |x + y| \leq |x \boxplus y|$.

Il s'ensuit

$$x \boxplus y = \Psi(x + y).$$

Démontrons la seconde affirmation.

La mantisse u comporte au plus p+1 chiffres. Donc, si $ub^g \notin S$, u comporte p+1 chiffres et le (p+1)ème chiffre est différent de zéro. Dès lors, u' = u et $\varphi((|u| - b^{-(p+1)})\, b^g) = \varphi(|ub^g|)$.

Les mantisses n' et n" vérifient

$$|n"| \leq |n'| < |n"| + b^{-(p+1)}.$$

Donc $|m| - |n"| - b^{-(p+1)} < |m| - |n'| \leq |m| - |n"|$.

L'application φ est monotone. Il s'ensuit

$$\varphi((|m| - |n"| - b^{-(p+1)})\, b^g) = \varphi(|ub^g|) \leq$$
$$\varphi(|x + y|) \leq \varphi(|ub^g|) = |x \boxplus y|.$$

Et finalement

$$x \boxplus y = \varphi(x + y).$$

c) q = 0 et e - f = 1.

Avant d'être additionné à x, le nombre $y = nb^{e-1}$ subit un "shifting" à droite d'un chiffre hexadécimal. Le $p^{ème}$ chiffre de n est perdu au cours de cette transformation.

Par conséquent $n''b^e = \varphi^*(nb^{e-1})$

Le nombre $u'b^e \in S$, donc

$$u'b^e = ub^g = \varphi(ub^g).$$

Finalement

$$x \boxplus y = x + \varphi^*(y).$$

d) $q = 0$ et $e - f > 1$.

La mantisse u' vérifie

$$b^{-2} < |u'| < 1 \quad \text{et} \quad ub^g \in S$$

Si $b^{-1} \leq |u'| < 1$ alors $u' = u$ et, en raisonnant comme nous l'avons fait pour démontrer la première affirmation en (b), nous obtenons

$$x \boxplus y = \Psi(x + y).$$

Si $b^{-2} < |u'| < b^{-1}$, alors $u = u'b \in T$. Donc

$$x \boxplus y = \Psi^*(x + y).$$

Finalement, nous pourrons énoncer le théorème suivant :

Théorème 8.

Soient x et y des nombres flottants notés $x = mb^e$ $y = nb^f$ avec $e \geq f$.

Si x et y sont de même signe, alors

$$x \boxplus y = \varphi(x + y).$$

Par contre, si x et y sont de signe contraire :

1) $q \neq 0$, alors

$$x \boxplus y = \varphi(x + y) \text{ ou } x \boxplus y = \Psi(x + y).$$

2) $q = 0$ et $e = f$, alors

$$x \boxplus y = x + y$$

$q = 0$ et $e - f = 1$, alors

$$x \boxplus y = x + \varphi^*(y)$$

$q = 0$ et $e - f > 1$, alors

$$x \boxplus y = \Psi(x + y) \quad \text{ou} \quad x \boxplus y = \Psi^{*}(x + y).$$

3.3.3. Propriétés de l'addition en simple longueur.

1) Le groupoïde $\{S, \boxplus\}$ ne respecte pas la relation d'ordre sur M par rapport à la loi +.

Considérons l'exemple suivant :

$x = 0.1\ 16^{0}$, $y = -0.543\ 16^{-5}$, $z = 0.FFF\ 16^{0}$,

$t = 0.ABE\ 16^{-5}$

$$x + y = 0.FFFFABD\ 16^{-1} < 0.FFFFABE\ 16^{-1} = z + t.$$

Par contre,

$$x \boxplus y = 0.FFFFAC\ 16^{-1} > 0.FFFFAB\ 16^{-1} = z \boxplus t.$$

2) Le groupoïde $\{S, \boxplus\}$ est optimal.

Soient x et $y \in S$. Posons $x \boxplus y = \square_{+}(x + y)$. Donc $\square_{+} = \Psi$ ou $\square_{+} = \Psi$. Or Ψ et Ψ sont des applications optimales. Dès lors, par le théorème 6, le groupoïde $\{S, \boxplus\}$ est optimal.

3) La loi $\boxplus$ est commutative dans S.

Ceci est une conséquence du théorème 6 car la loi + est commutative dans $\mathbb{R}$.

4) Le groupoïde $\{S, \boxplus\}$ n'est pas un groupoïde dirigé vers zéro.

Si $x = 0.1\ 16^{0}$ et $y = -0.543\ 16^{-5}$,

$$x + y = 0.FFFFABD\ 16^{-1} \geq 0 \quad \text{et}$$

$$x \boxplus y = 0.FFFFAC\ 16^{-1} > x + y$$

$\{S, \boxplus\}$ n'est donc pas un groupoïde dirigé vers zéro.

5) Quel que soit $x \in S$, $x \boxplus 0 = 0 \boxplus x = x$.

Le groupoïde $\{S, \boxplus\}$ est optimal. Donc, par le théorème 7, 0 est neutre de S pour la loi $\boxplus$.

3.3.4. Propriétés de l'addition en double longueur.

1) Le groupoïde $\{S, \boxplus\}$ ne respecte pas la relation d'ordre sur M par rapport à la loi +.

Prenons par exemple

$x = 0.12345678912345\ 16^{0}$,

$y = -0.19\ 16^{-4}$,

$z = 0.1234567891238\ 16^{0}$,

$t = 0.38\ 16^{-4}$.

$x + y = 0.123456789123437 < 0.12345678912344 = z + t.$

Par contre,

$x \boxplus y = 0.12345678912344 > 0.12345678912343 = z \boxplus t.$

Cet exemple montre bien que $\{S, \boxplus\}$ ne respecte pas la structure d'ordre sur M par rapport à l'addition dans $\mathbb{R}$.

2) Le groupoïde $\{S, \boxplus\}$ n'est pas optimal.

Considérons les nombres suivants :

$x = 0.1\ 16^{1}$ et $y = -0.1\ 16^{1} + b^{-p}$. x et $y \in S$.

Le nombre $x + y = b^{-p} \in S$. Pourtant,

$x \boxplus y = b^{-(p-1)} \neq x + y.$

Donc le groupoïde $\{S, \boxplus\}$ n'est pas optimal.

3) La loi $\boxplus$ est commutative dans S.

4) Le groupoïde $\{S, \boxplus\}$ n'est pas un groupoïde dirigé vers zéro.

5) Quel que soit $x \in S$, $x \boxplus 0 = 0 \boxplus x = x$.

La représentation flottante normalisée de zéro est

$$+0.00\ldots0\ b^{e_1}.$$

Soit $x = mb^e \in S$. Donc $e \geqslant e_1$.

Si $e = e_1$, alors $x \boxplus 0 = x + 0 = x$.

Si $e = e_1 + 1$, alors $x \boxplus 0 = x + \varphi^*(0) = x + 0$.

Si $e - e_1 > 1$, alors $u = m + 0$ et $b^{-1} \leqslant |u| < 1$.

Par conséquent, $x \boxplus 0 = \Psi(x + 0) = x$.

Finalement, zéro est un neutre de S pour la loi $\boxplus$.

3.3.5. Etude de l'erreur relative.

Si l'addition s'effectue avec l'aide d'un chiffre de garde, quels que soient $x,y \in S$,

$$x \boxplus y = (x + y)\ (1 + \varepsilon) \text{ avec } |\varepsilon| < b^{1-p}. \qquad (9)$$

En effet, $q \neq 0$ implique

$$x \boxplus y = \Phi(x + y) \quad \text{ou} \quad x \boxplus y = \Psi(x + y).$$

Par conséquent, $x \boxplus y = (x + y)\ (1 + \varepsilon)$ et ε vérifie : $|\varepsilon| < b^{1-p}$.

Si l'addition s'effectue sans l'aide d'un chiffre de garde, quels que soient x et $y \in S$,

$$x \boxplus y = x(1 + \varepsilon_1) + y(1 + \varepsilon_2)$$
$$\text{et } |\varepsilon_1|, |\varepsilon_2| < b^{2-p}. \qquad (10)$$

Le nombre q vaut zéro. Si $x \boxplus y = \Phi(x + y)$ ou $x \boxplus y = \Psi(x + y)$, alors

$$x \boxplus y = (x + y)\ (1 + \varepsilon) \quad \text{avec } |\varepsilon| < b^{1-p}.$$

Dans le cas $x \boxplus y = \Psi^*(x + y)$,

$$x \boxplus y = (x + y)\ (1 + \varepsilon) \text{ avec } |\varepsilon| < b^{2-p}.$$

Et enfin, si $x \boxplus y = x + \Phi^*(y)$,

$$x \boxplus y = x + y(1 + \varepsilon) \quad \text{avec } |\varepsilon| < b^{2-p}.$$

Par conséquent, quels que soient x et $y \in S$, nous pourrons mettre $x \boxplus y$ sous la forme

$$x \boxplus y = x(1 + \varepsilon_1) + y(1 + \varepsilon_2)$$
$$\text{et } |\varepsilon_1|, |\varepsilon_2| < b^{2-p}.$$

Remarque : Si $q = 0$, nous n'avons plus une borne intéressante pour ε tel que

$$x \boxplus y = (x + y)\ (1 + \varepsilon).$$

Si nous prenons l'exemple

$$x = 0.1\ 16^{1} \quad \text{et} \quad y = -0.1\ 16^{1} + b^{-p},$$

les résultats sont les suivants :

$$x \boxplus y = b^{-(p-1)}$$

$$x + y = b^{-p}.$$

L'erreur relative ε vaut donc

$$\frac{b^{-(p-1)} - b^{-p}}{b^{-p}} = b - 1 = 15.$$

3.4. Multiplication en virgule flottante.

3.4.1. Description des opérations sur l'ordinateur Siemens.

1) Multiplication de deux nombres flottants de longueur simple.

Soient x et y $\in$ S(16,6,-64,63).

1ère étape.

Les caractéristiques sont additionnées et la somme, moins 64, forme la caractéristique intermédiaire.

2ème étape.

Les mantisses sont multipliées. La mantisse du produit intermédiaire occupe douze chiffres hexadécimaux.

3ème étape.

Le produit intermédiaire est normalisé et sa mantisse est tronquée à six chiffres hexadécimaux.

Exemple.

$$0.111111\ 16^{0} \boxdot 0.111111\ 16^{0} = 0.123456\ 16^{-1}$$

En effet, le produit intermédiaire est

$$0.012345654321\ 16^{0}.$$

Après normalisation, seuls les six premiers chiffres de la mantisse sont pris en considération pour former la mantisse du résultat final.

2) Multiplication de deux nombres flottants de longueur double.

Soient x et y des éléments de S(16,14,-64,63).

1ère étape.

Les caractéristiques sont additionnées. La somme, moins 64, forme la caractéristique intermédiaire.

2ème étape.

Les mantisses sont multipliées. La mantisse du produit intermédiaire consiste en les quatorze premiers chiffres du résultat exact.

3ème étape.

Le résultat intermédiaire est normalisé.

Exemple.

$$0.1\ 16^{0} \boxdot 0.FFFFFFFFFFFFFF\ 16^{-1} = 0.FFFFFFFFFFFFF0\ 16^{-2}$$

En effet, le résultat intermédiaire est

$$0.0FFFFFFFFFFFFF\ 16^{-1}.$$

Après normalisation, nous obtenons le résultat final.

3.4.2. Etude du processus de la multiplication.

Cette étude nécessite l'introduction d'un nombre entier q. Celui-ci vaut 0 ou p suivant que la mantisse du résultat intermédiaire comporte p ou 2p chiffres.

Soient x et y deux nombres flottants signés,

$$x = mb^{e} \quad \text{et} \quad y = nb^{f}.$$

Si x = 0 ou y = 0, la mantisse du résultat intermédiaire vaut zéro. Après normalisation,

$$x \boxdot y = +0.00\ldots0\ b^{e_1}.$$

Considérons à présent le cas où x et y sont différents de zéro. Nous désignerons par u' la mantisse qui consiste en les p+q premiers chiffres de m.n et par u la mantisse obtenue après normalisation de u'.

Le nombre entier g est défini par

$$x \boxdot y = \varphi(ub^g).$$

Les mantisses m et n vérifient

$$b^{-1} \leq |m| < 1 \quad \text{et} \quad b^{-1} \leq |n| < 1.$$

Par conséquent, $b^{-2} \leq |u'| < 1$.

Supposons $b^{-1} \leq |u'| < 1$. Dans ce cas u' = u et les p premiers chiffres de u sont aussi les p premiers chiffres de m.n.

Par conséquent

$$x \boxdot y = \varphi(x \cdot y).$$

Considérons à présent le cas $b^{-2} \leq |u'| < b^{-1}$ et $q \neq 0$. Les p premiers chiffres de u = bu' sont les p premiers chiffres de m . n car $q \neq 0$.

Dès lors

$$x \boxdot y = \varphi(x \cdot y).$$

Finalement, dans le cas $b^{-2} \leq |u'| < b^{-1}$ et q = 0, la mantisse u = bu' consiste en les p-1 premiers chiffres de m . n. Donc $ub^g \in T$.

Il s'ensuit

$$x \boxdot y = \varphi^*(x \cdot y).$$

Nous résumons les résultats dans le théorème suivant.

Théorème 9.

Soient x et y deux nombres flottants signés.
Si $q \neq 0$, $x \boxdot y = \varphi(x \cdot y)$.
Si q = 0, $x \boxdot y = \varphi(x \cdot y)$ ou $x \boxdot y = \varphi^*(x \cdot y)$.

3.4.3. Propriétés de la multiplication en simple longueur.

1) Le groupoïde $\{S, \boxdot\}$ respecte la structure d'ordre sur M par rapport à la loi $\cdot$.

2) le groupoïde $\{S, \boxdot\}$ est optimal.

3) La loi $\boxdot$ est commutative dans S.

4) Le groupoïde $\{S, \boxdot\}$ est un groupoïde dirigé vers zéro.

5) Quel que soit $x \in S$, $x \boxdot 0.1\ b = 0.1\ b \boxdot x = x$.

Ces propriétés sont les conséquences des théorèmes 6, 7 et 9.

3.4.4. Multiplication de deux nombres de double longueur.

1) Le groupoïde $\{S, \boxdot\}$ ne respecte pas la relation d'ordre sur M par rapport à la loi $\cdot$.

Considérons les nombres suivants :

$x = 0.1111111111111\ 16^{0}$,

$y = 0.F111111111111F\ 16^{-1}$,

$z = 0.100000000000000\ 16^{1}$,

$t = 0.10123456789ABF\ 16^{-1}$.

$x \cdot y = 0.10123456789ABDBC...$

$< 0.10123456789ABF = z \cdot t.$

$x \boxdot y = 0.10123456789ABD >$

$0.10123456789ABO = z \boxdot t.$

2) Le groupoïde $\{S, \boxdot\}$ n'est pas optimal.

Prenons pas exemple

$x = 0.FFFFFFFFFFFFF\ 16^{0}$,

$y = 0.1\ 16^{1}$.

$x \cdot y = x \in S$ et

$x \boxdot y = 0.FFFFFFFFFFFFO\ 16^{0} \neq x \cdot y.$

3) La loi $\boxdot$ est commutative dans S.

4) Le groupoïde $\{S, \boxdot\}$ est un groupoïde dirigé vers zéro.

Quels que soient x et $y \in S$,

$x \boxdot y = \varphi(x \cdot y)$ ou $x \boxdot y = \varphi^{*}(x \cdot y)$.

Par conséquent, si $x \cdot y \geqslant 0$,

$x \boxdot y \leqslant x \cdot y$

et si $x \cdot y < 0$,

$x \boxdot y \geqslant x \cdot y$

5) Il n'existe pas d'élément neutre dans S pour la loi $\boxdot$.

Si nous reprenons l'exemple

$x = 0.FFFFFFFFFFFFFFF\ 16^{0}$

$e = 0.1\ 16^{1}$

$x \boxdot e \neq x$

Le nombre e était le neutre pour $\cdot$ et pour $\boxdot$ en simple longueur.

3.4.5. Erreur relative.

Si la mantisse du résultat intermédiaire comporte au moins p+1 chiffres, quels que soient $x, y \in S$,

$$x \boxdot y = (x \cdot y)(1 + \varepsilon) \quad \text{avec } |\varepsilon| < b^{1-p}. \qquad (11)$$

En effet, $q \neq 0$ implique

$x \boxdot y = \varphi(x \cdot y)$.

Par conséquent, $x \boxdot y = (x \cdot y)(1 + \varepsilon)$ et ε vérifie $|\varepsilon| < b^{1-p}$.

Si la mantisse du résultat intermédiaire comporte p chiffres, quels que soient $x,y \in S$,

$$x \boxdot y = (x \cdot y)(1+\varepsilon) \quad \text{avec } |\varepsilon| < b^{2-p}. \tag{12}$$

Si $q = 0$, on a $x \boxdot y = \varphi(x \cdot y)$ ou $x \boxdot y = \varphi^*(x \cdot y)$. Dans le premier cas,

$$x \boxdot y = (x \cdot y)(1+\varepsilon) \quad \text{avec } |\varepsilon| < b^{1-p}.$$

Dans le second cas,

$$x \boxdot y = (x \cdot y)(1+\varepsilon) \quad \text{avec } |\varepsilon| < b^{2-p}.$$

Par conséquent, quels que soient $x,y \in S$,

$$x \boxdot y = (x \cdot y)(1+\varepsilon) \quad \text{avec } |\varepsilon| < b^{2-p}.$$

3.5. La division en virgule flottante.

3.5.1. Description de l'opération sur l'ordinateur Siemens.

Soient x et $y \in S(16,p,-64,63)$ où $p = 6$ ou 14.

1ère étape.

Les caractéristiques du diviseur et du dividende sont soustraites. La différence, augmentée de 64, forme la caractéristique du résultat intermédiaire.

2ème étape.

Les mantisses du dividende et du diviseur sont divisées pour donner la mantisse du quotient intermédaire. Celle-ci comprend les six premiers chiffres hexadécimaux du résultat exact en simple longueur, les quatorze premiers en double longueur, plus un report possible.

3ème étape.

Le résultat intermédiaire est normalisé. Sa mantisse est tronquée à six chiffres en simple longueur et à quatorze chiffres en double longueur.

3.5.2. Etude du processus de la division.

Soient x et y des nombres flottants signés

$x = mb^e$ et $y = nb^f \neq 0$.

Désignons par u' la mantisse du quotient intermédiaire et par u la mantisse obtenue après normalisation de u'. Le nombre g sera celui qui vérifie

$$x \boxslash y = ub^g$$

Si $x = 0$, alors $x \boxslash y = x / y = 0$.

Prenons le cas $x \neq 0$. Les mantisses m et n vérifient

$$b^{-1} \leq |m| < 1 \quad \text{et} \quad b^{-1} \leq |n| < 1.$$

Donc, $b^{-1} < |u'| < b$.

Supposons $b^{-1} \leq |u'| < 1$. Alors $u = u'$ et u consiste en les p premiers chiffres de la mantisse du résultat exact.

Par conséquent

$$x \boxslash y = \varphi(x / y).$$

Dans l'autre cas, c'est-à-dire $1 \leq |u'| < b$, la mantisse $u = u'b$ consiste en les $p+1$ premiers chiffres de la mantisse du résultat exact et

$$x \boxslash y = \varphi(x / y) = \varphi(ub^{g}).$$

Nous résumons nos résultats dans le théorème suivant :

Théorème 10.

Soient x et y deux nombres flottants signés et $y \neq 0$. La division flottante vérifie

$$x \boxslash y = \varphi(x / y).$$

3.5.3. Propriétés de la division en virgule flottante.

1) Le groupoïde $\{S, \boxslash\}$ respecte la structure d'ordre sur M par rapport à la loi /.

2) Le groupoïde $\{S, \boxslash\}$ est optimal.

3) Le groupoïde $\{S, \boxslash\}$ est un groupoïde dirigé vers zéro.

4) Le nombre $0.1\ b^{1}$ est neutre à droite de S pour la loi $\boxslash$.

3.5.4. Etude de l'erreur.

Quels que soient $x, y \in S$ tels que $y \neq 0$,

$$x \boxslash y = (x / y)(1 + \varepsilon) \quad \text{avec } |\varepsilon| < b^{1-p}. \tag{13}$$

Ceci est une conséquence immédiate du théorème 10.

4. LES LOIS ALGEBRIQUES.

Soient x, y et z $\in$ S = S(16,p,-64,63). Posons p' = p-1 si p = 6 et p' = p-2 si p = 14.

4.1. Associativité de la loi additive.

La loi + est associative dans $\mathbb{R}$ en ce sens que

$$\forall x,y,z \in \mathbb{R} \quad (x + y) + z = x + (y + z).$$

Par contre, la loi $\boxplus$ n'est pas associative dans S. En effet, considérons les résultats (9) et (10) obtenus dans la section 3.3. Nous pouvons en déduire

$$x \boxplus (y \boxplus z) = x(1 + \varepsilon_1) + y(1 + \varepsilon_2) + z(1 + \varepsilon_3),$$

où ε_1, ε_2 et ε_3 vérifient

$$1 - b^{-p'} < 1 + \varepsilon_1 < 1 + b^{-p'}$$

$$(1 - b^{-p'})^2 < 1 + \varepsilon_i < (1 + b^{-p'})^2 \qquad i = 2,3.$$

Mais

$$(x \boxplus y) \boxplus z = x(1 + \varepsilon'_1) + y(1 + \varepsilon'_2) + z(1 + \varepsilon'_3)$$

où ε'_1, ε'_2 et ε'_3 vérifient

$$(1 - b^{-p'})^2 < 1 + \varepsilon'_i < (1 + b^{-p'})^2 \qquad i = 1,2$$

$$1 - b^{-p'} < 1 + \varepsilon'_3 < 1 + b^{-p'}.$$

Par conséquent, les erreurs associées aux nombres x, y et z ont des bornes qui dépendent de l'ordre dans lequel les additions sont effectuées.

Illustrons la non validité de l'associativité de $\boxplus$ dans S par un exemple très simple. Soient x, y et z $\in$ S(16,6,-64,63).

$$x = -0.534591$$

$$y = +0.537893$$

$$z = +0.56789A\ 16^{-2}.$$

Nous avons alors

$$x \boxplus (y \boxplus z) = 0.897A\ 16^{-2}.$$

Mais

$$(x \boxplus y) \boxplus z = 0.897A94\ 16^{-2}$$

qui est aussi le résultat exact de l'addition.

4.2. Associativité de la loi multiplicative.

La loi $\cdot$ est associative dans $\mathbb{R}$. C'est-à-dire $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

Nous allons montrer que la loi $\boxdot$ est pratiquement associative dans S. Considérons les résultats (11) et (12) obtenus dans la section 3.4. Nous pouvons en déduire

$$(x \boxdot y) \boxdot z = \left[(x \cdot y) \cdot z\right] (1 + \varepsilon_1)$$

avec

$$(1 - b^{-p'})^2 < 1 + \varepsilon_1 < (1 + b^{-p'})^2$$

et

$$x \boxdot (y \boxdot z) = \left[x \cdot (y \cdot z)\right] (1 + \varepsilon_2)$$

avec

$$(1 - b^{-p'})^2 < 1 + \varepsilon_2 < (1 + b^{-p'})^2.$$

Par conséquent

$$(x \boxdot y) \boxdot z = x \boxdot (y \boxdot z) \frac{(1 + \varepsilon_1)}{(1 + \varepsilon_2)}.$$

Pour ε_2 petit, on a $\dfrac{(1 + \varepsilon_1)}{(1 + \varepsilon_2)} \approx (1 + \varepsilon_1)(1 - \varepsilon_2) = (1 + \eta)$.

Finalement,

$$(x \boxdot y) \boxdot z = x \boxdot (y \boxdot z)(1 + \eta)$$

où η vérifie

$$(1 - b^{-p'})^4 < 1 + \eta < (1 + b^{-p'})^4.$$

Exemple.

Soient x, y et z $\in$ S(16,6,-64,63) où

$x = 0.FE$

$y = 0.1000FF\ 16^1$

$z = 0.101006\ 16^1$.

Nous avons alors

$$(x \boxdot y) \boxdot z = \text{O.FFOE3F}$$

et

$$x \boxdot (y \boxdot z) = \text{O.FFOE2F.}$$

Le rapport $\dfrac{(x \boxdot y) \boxdot z}{x \boxdot (y \boxdot z)} = (1 + \eta)$ vérifie bien

$$(1 - 16^{-5})^4 < 1 + \frac{1}{\text{FFOE2.F}} < (1 + 16^{-5})^4.$$

4.3. Distributivité de la loi multiplicative par rapport à la loi additive.

La multiplication dans $\mathbb{R}$ est distributive par rapport à l'addition. Donc, pour tout $x, y, z \in \mathbb{R}$,

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z).$$

Cette propriété n'est pas vérifiée pour $\boxdot$ et $\boxplus$ dans S. Considérons d'abord le cas

$$(y \boxplus z) = (y + z)(1 + \varepsilon) \qquad \text{(voir 3.3.(9))}.$$

Alors,

$$x \boxdot (y \boxplus z) = x \cdot (y + z)(1 + \eta)$$

où η vérifie

$$(1 - b^{-p'})^2 < 1 + \eta < (1 + b^{-p'})^2.$$

Par contre,

$$(x \boxdot y) \boxplus (x \boxdot z) = (x \cdot y)(1 + \varepsilon_1) + (x \cdot z)(1 + \varepsilon_2)$$

avec

$$(1 - b^{-p'})^2 < 1 + \varepsilon_i < (1 + b^{-p'})^2 \qquad i = 1,2.$$

Exemple.

Soient $x, y, z \in S(16,6,-64,63)$, $\boxplus$ et $\boxdot$ qui vérifient (9) et (11). Si

$$x = 0.111111$$
$$y = -0.1$$
$$z = 0.10000E,$$

nous avons

$$x \boxdot (y \boxplus z) = 0.EEEEEE\ 16^{-6}$$

qui est le résultat exact alors que

$$(x \boxdot y) \boxplus (x \boxdot z) = 0.E\ 16^{-6}.$$

Si la loi $\boxplus$ vérifie

$$(y \boxplus z) = y(1 + \alpha) + z(1 + \beta) \qquad \text{(voir 3.3.(10))},$$

alors

$$x \boxdot (y \boxplus z) = x \cdot y\,(1 + \varepsilon_1) + x \cdot z\,(1 + \varepsilon_2)$$

où

$$(1 - b^{-p'})^2 < 1 + \varepsilon_i < (1 + b^{-p'})^2 \qquad i=1,2$$

et

$$(x \boxdot y) \boxplus (x \boxdot z) = x \cdot y\,(1 + \varepsilon'_1) + x \cdot z\,(1 + \varepsilon'_2)$$

où

$$(1 - b^{-p'})^2 < 1 + \varepsilon'_i < (1 + B^{-p'})^2.$$

4.4. Simplification à gauche et à droite.

Si x, y et z sont des nombres réels et $x \neq 0$,

$$x \cdot y = x \cdot z \Longrightarrow y = z.$$

Cette propriété est quasiment vérifiée dans S.

Soient x, y et z des éléments de S, $x \neq 0$ et $x \boxdot y = x \boxdot z$. Alors, il existe ε_1 et ε_2 tels que

$$x \boxdot y = (x \cdot y)(1 + \varepsilon_1) = (x \cdot z)(1 + \varepsilon_2) = x \boxdot z,$$

et

$$1 - b^{-p'} < 1 + \varepsilon_i < 1 + b^{-p'} \qquad i = 1,2.$$

Par conséquent,

$$y = z(1 + \eta)$$

où η vérifie

$$(1 - b^{-p'})^2 < 1 + \eta < (1 + b^{-p'})^2.$$

Exemple.

Soient x, y et $z \in S(16,6,-64,63)$ et $\boxdot$ qui vérifie la relation (11),

$$x = 0.11\ 16^1$$
$$y = 0.FF$$
$$z = 0.FF000F.$$

Nous avons

$$x \boxdot y = x \boxdot z = 0.10EF\ 16^1$$

et cependant, $y \neq z$.

Le quotient $\frac{y}{z}$ vérifie

$$(1 - 16^{-5})^2 < \frac{y}{z} = 1 - \frac{1}{110001} < (1 + 16^{-5})^2.$$

4.5. Inverse pour la loi multiplicative.

Dans $\mathbb{R}$, si $x \neq 0$, $x \cdot (1 / x) = 1$.

Qu'en est-il de cette relation dans l'ensemble S ?
Si $x \neq 0$ et $x \in S$,

$$x \boxdot (1 \boxslash x) = x \cdot (1 / x)(1 + \varepsilon)(1 + \eta)$$
$$= (1 + \varepsilon)(1 + \eta),$$

où ε et η vérifient

$$1 - b^{-p'} < 1 + \varepsilon < 1 + b^{-p'} \qquad \text{(voir 3.4.(11) et (12))}$$
$$1 - b^{1-p} < 1 + \eta < 1 + b^{1-p} \qquad \text{(voir 3.5.(13))}.$$

En particulier, si la loi $\boxdot$ vérifie la relation (11),

$$x \boxdot (1 \boxslash x) = (1 + \alpha)$$

avec

$$(1 - b^{1-p})^2 < 1 + \alpha < (1 + b^{1-p})^2.$$

Exemple.

Considérons un nombre $x \in S(16,6,-64,63)$, $\boxdot$ qui vérifie la relation (11) et $x = 0.AB5938$.

$$x \boxdot (1 \boxslash x) = 0.FFFFFE \neq 1,$$

mais

$$(1 - 16^{-5})^2 < 1 - \frac{2}{16^5} < (1 + 16^{-5})^2.$$

On peut généraliser cette propriété de la manière suivante : dans $\mathbb{R}$, si $x \neq 0$,

$$x \cdot (y / x) = y.$$

Cette propriété est quasiment vérifée dans S. En effet,

$$x \boxdot (y \boxslash x) = y\ (1 + \varepsilon)(1 + \eta)$$

où ε et η vérifient

$$1 - b^{-p'} < 1 + \varepsilon < 1 + b^{-p'}$$
$$1 - b^{1-p} < 1 + \eta < 1 + b^{1-p}.$$

Dans le cas particulier où $p' = p-1$,

$$x \boxdot (y \boxslash x) = y\,(1 + \varepsilon)$$

où

$$(1 - b^{-p'})^2 < 1 + \varepsilon < (1 + b^{-p'})^2.$$

Exemple.

Si x = O.FA6BC3 et y = O.AB5938, qui sont des éléments de S(16,6,-64,63), et si $\boxdot$ et $\boxslash$ vérifient les relations (11) et (13),

$$x \boxdot (y \boxslash x) = \text{O.AB5937} \neq y.$$

Chap. II - EFFET DES ERREURS D'ARRONDI SUR UN ALGORITHME

1. INTRODUCTION.

Dans ce deuxième chapitre, nous allons effectuer une analyse d'erreur à propos du problème suivant.

Nous disposons de deux polynômes factorisés :

$$p_1(x) = k_1 \prod_{i=1}^{n} (x - a_i) \qquad (14)$$

$$p_2(x) = k_2 \prod_{i=1}^{m} (x - b_i) \qquad (15)$$

Nous désirons calculer les zéros du polynôme

$$p(x) = p_1(x) + p_2(x) \qquad (16)$$

L'intérêt de ce problème est assez clair : en voici deux aspects :

1.1. Synthèse des filtres [10].

Dans la synthèse des filtres, nous sommes amenés à manipuler des polynômes et, bien souvent, nous nous trouvons en face d'un problème similaire à celui exposé ci-dessus.

1.2. Calcul des racines de polynômes orthogonaux.

Nul n'ignore l'importance des fonctions orthogonales dans le problème de la quadrature de Gauss par exemple, ou encore le rôle des polynômes de Tchebycheff, Legendre,... Pour ces polynômes, nous disposons bien souvent d'une relation de récurrence qui va nous permettre de calculer leurs racines de proche en proche.

Exemple.

Pour les polynômes de Legendre, nous disposons des deux premiers polynômes

$$p_0(x) = 1$$

$$p_1(x) = x$$

et de la relation de récurrence

$$(m + 1)\, p_{m+1}(x) = (2m + 1)\, p_m(x) - m\, p_{m-1}(x).$$

Dans les deux cas, nous nous trouvons en face d'un problème de précision numérique. Nous verrons que la méthode usuelle qui consiste à calculer les coefficients, puis à factoriser le polynôme, nécessite un nombre considérable de chiffres. Or, nous sommes limités par la représentation de ces nombres sur machine. Pour éviter ces difficultés, une deuxième méthode a été proposée : la méthode produit.

Nous étudierons tout d'abord le comportement des racines lorsque les polynômes sont définis par les coefficients, en utilisant la théorie des fonctions algébriques. Nous effectuerons par après une analyse de la méthode de Newton-Raphson, utilisée pour calculer les racines et, enfin, nous verrons en quoi la méthode produit permet d'améliorer la précision numérique des résultats par rapport à la méthode usuelle. Nous terminerons par quelques exemples numériques.

2. METHODE USUELLE.

2.1. Principe de la méthode.

Dans une première étape, nous allons calculer les coefficients des polynômes $p_1(x)$ et $p_2(x)$ définis en (14) et (15).

Ces coefficients sont des fonctions symétriques des racines. A partir de (16), nous pouvons alors facilement calculer les coefficients du polynôme $p(x)$.

Dans la seconde étape, consistant en la factorisation du polynôme $p(x)$, nous allons utiliser la méthode de Newton-Raphson, généralisée dans le cas complexe pour obtenir toutes les racines de ce polynôme.

Nous allons donc analyser la sensibilité des racines vis-à-vis de changements des coefficients, étude faite par Hille [11] et par Wilkinson [1]. Ces changements des coefficients proviennent du calcul des coefficients et des erreurs d'approximation dans la méthode de Newton-Raphson : dans ces deux cas, nous allons essayer d'estimer ces erreurs.

2.2. Les fonctions algébriques.

2.2.1. Définitions.

Soit un polynôme à deux variables complexes :

$$F(z,w) = p_0(z)\, w^n + p_1(z)\, w^{n-1} + \ldots + p_n(z) \qquad (17)$$

où

$$p_j(z) = \sum_{k=0}^{m_j} a_{j,k}\, z^k \qquad (j = 0,\ldots,n).$$

Pour une valeur fixée de z, l'équation en w

$$F(z,w) = 0 \qquad (18)$$

admet normalement n solutions distinctes finies

$$w_1(z),\ w_2(z),\ldots,w_n(z).$$

Ce sont les différents éléments de la fonction algébrique que nous allons étudier.

Il existe cependant certaines valeurs particulières de z pour lesquelles l'équation en w (18) n'admet pas n solutions distinctes finies. Ces valeurs, appelées points exceptionnels de F(z,w) sont de trois types :

- le point à l'infini
- les racines de $p_0(z)$
- les valeurs de z pour lesquelles le polynôme en w (17) admet des racines de multiplicité supérieure à 1.

Un point non exceptionnel sera dit ordinaire.

Nous allons étudier le comportement de la fonction $w_i(z)$ $(i=1,\ldots,n)$ qui coïncide en z_0 avec w_0, au voisinage de z_0 dans le cas où w_0 est une racine finie de multiplicité égale ou supérieure à 1 du polynôme $F(z_0,w)$.

2.2.2. Rappel sur les fonctions méromorphes.

Soit une fonction f(z) à valeur et variable complexes. Nous dirons que f(z) est méromorphe en un domaine D si elle n'admet pas d'autres singularités que des pôles d'ordre (de multiplicité) fini(e) en D.

Rappelons deux résultats que nous utiliserons ultérieurement : soient C une courbe simple, fermée, orientée positivement et C^* le compact délimité par C.

1) Principe de l'argument généralisé.

Si f(z) est méromorphe sur C^* et n'admet ni pôle ni zéro sur C, si g(z) est holomorphe sur C^*, alors

$$\frac{1}{2i\pi}\int_C g(z)\,\frac{f'(t)}{f(t)}\,dt = \sum_{j=1}^{m} g(a_j)\,m_j - \sum_{j=1}^{n} g(b_j)\,n_j$$

où a_j (j=1,...,m) sont les zéros de f(z), de multiplicité m_j situés dans C^*, et b_j (j=1,...,n) sont les pôles de f(z), de multiplicité n_j situés dans C^*.

2) Théorème de Rouché.

Si f(z) est méromorphe sur C^* et peut s'écrire comme la somme de deux fonctions méromorphes sur C^*

$$f(z) = g(z) + h(z) \qquad \forall z \in C^*,$$

si g(z) et f(z) n'admettent ni pôle ni zéro sur C, si, en plus

$$|g(z)| > |h(z)| \qquad \forall z \in C,$$

alors la différence entre le nombre de zéros et le nombre de pôles, comptés avec leur multiplicité, appartenant à C^*, est identique pour g(z) et f(z).

2.2.3. Fonctions inverses.

Nous voulons étudier les solutions de l'équation

$$f(z) = w \tag{19}$$

où f(z) est une fonction holomorphe sur $|z| < R$.

Théorème 11.

Si f(z) satisfait aux conditions

$$f(0) = 0$$
$$f'(0) \neq 0$$
$$f(z) \neq 0 \qquad 0 < |z| \leq r < R$$

pour une circonférence C de centre O et de rayon $\rho < r$, la fonction

$$g(w) = \frac{1}{2i\pi} \int_C t \frac{f'(t)}{f(t)-w} \, dt$$

est holomorphe pour $|w| < m = \min_{\theta} |f(\rho e^{i\theta})|$.

Pour de telles valeurs de w, $z = g(w)$ est la solution unique de (19).

Preuve

Considérons les deux fonctions holomorphes $f(z)$ et $f(z)-w$ où $|w| < m$.

Sur la circonférence C, nous avons $|f(z)| \geq m > |-w|$ et par application du théorème de Rouché, nous savons que les fonctions $f(z)$ et $f(z)-w$ ont un même nombre de zéros sur $0 \leq |z| \leq \rho$.

Comme $f(z)$ y admet un zéro simple, nous concluons que l'équation (19) admet une sdution unique notée $g(w)$ située dans $0 \leq |z| \leq \rho$. Appliquons maintenant le principe de l'argument généralisé à la fonction $f(t)-w$; nous obtenons

$$\frac{1}{2i\pi} \int_C t \frac{f'(t)}{f(t)-w} \, dt = g(w).$$

Démontrons maintenant qu'il s'agit d'une fonction holomorphe. Nous avons l'égalité

$$\frac{1}{f(t)-w} = \frac{1}{f(t)} + \frac{w}{(f(t))^2} + \frac{w^2}{(f(t))^3} + \ldots,$$

réalisée pour $|w| \leq m(1-\delta)$ $\quad (0 < \delta < 1)$ et $t \in C$.

Si nous multiplions cette série par la fonction bornée $t\,f'(t)$ et si nous intégrons terme à terme, nous obtenons

$$g(w) = \sum_{n=0}^{\infty} w^n \frac{1}{2i\pi} \int_C t \frac{f'(t)}{f(t)} \, dt$$

■

Théorème 12.

Supposons que

$$f(0) = f'(0) = \ldots = f^{(k-1)}(0) = 0$$

$$f^{(k)}(0) \neq 0$$

$$f(z) \neq 0 \qquad 0 < |z| < r \leq R$$

Alors, pour de petites valeurs de $|w|$, l'équation (19) admet k racines

$$z_1(w),\ z_2(w),\ldots,\ z_k(w)$$

qui tendent vers 0 avec w. Il existe une fonction $g(W)$, holomorphe pour $|W|$ assez petit, telle que les k déterminations de $g(w^{1/k})$ représentent les solutions de (19) dans un certain ordre. Si w décrit un circuit autour de l'origine, ces solutions sont permutées de manière cyclique.

Preuve

Soit C' une circonférence de centre 0 et de rayon $\rho < r$, soit $m = \min_{\theta} |f(\rho e^{i\theta})|$.

Pour $|w| < m$, les fonctions $f(z)$ et $f(z)-w$ ont un nombre identique de zéros sur $0 \leq |z| \leq \rho$. La fonction $f(z)-w$ admet donc k zéros notés $z_1(w),\ldots,z_k(w)$. En w=0; elles coïncident avec l'origine.

D'autre part, il existe a tel que $k!a^k = f^{(k)}(0)$. Soit W tel que $w = W^k$. Nous savons que

$$f(z) = (az)^k \left[1 + \sum_{n=1}^{\infty} b_n z^n\right]$$

et que la fonction $\left[1 + \sum_{n=1}^{\infty} b_n z^n\right]$ ne s'annule pas sur $0 \leq |z| \leq r$.

Soit une famille $(c_p)_{p=1}^{\infty}$ vérifiant

$$\left[1 + \sum_{n=0}^{\infty} b_n z^n\right]^{1/k} = 1 + \sum_{p=1}^{\infty} c_p z^p$$

et considérons l'équation

$$F(z) = az\left[1 + \sum_{p=1}^{\infty} c_p z^p\right] = W.$$

Par application du théorème précédent, nous savons qu'il existe une fonction $g(W)$, holomorphe pour $|W|$ assez petit, qu'il existe une circonférence C centrée en O et de rayon assez petit, telle que

$$g(W) = \frac{1}{2i\pi} \int_C t \frac{f'(t)}{f(t)-W} dt$$

$g(W)$ vérifie

$$F(g(W)) = W$$

$$(F(g(W)))^k = W^k = w$$

$$f(g(w^{1/k})) = w \qquad (20)$$

Or, la quantité a peut aussi bien être remplacée par ηa avec $\eta^k = 1$; W peut être remplacé par ηW et $w^{1/k}$ par $\eta w^{1/k}$: l'équation (20) demeure valable.

Les k zéros $z_j(w)$ peuvent dès lors se représenter de la manière suivante :

$$z_j(w) = g(\omega^j w^{1/k}) \quad \text{où} \quad \omega = e^{2i\pi/k}$$
$$0 \leq \arg(w^{1/k}) < \frac{2\pi}{k} \quad j=1,\ldots,k$$

Dans cette représentation, les zéros sont permutés de manière cyclique, lorsque w parcourt un circuit autour de l'origine. ∎

2.2.4. Fonction implicite.

Théorème 13 (de la fonction implicite).

Supposons que la fonction $F(z,w)$ soit de la forme

$$F(z,w) = \sum_{m=0}^{\infty} \sum_{n=0}^{\infty} a_{m,n} z^m w^n$$

avec $a_{0,0} = 0$

$a_{0,1} \neq 0$

et que cette double série soit absolument convergente pour $|z| \leq R_1$, $|w| \leq R_2$.

Alors, il existe une fonction $f(z)$, unique, holomorphe en un certain voisinage $|z| < \rho$ tel que

$$f(0) = 0$$

$$F(z,f(z)) = 0 \qquad |z| < \rho$$

Cette fonction est représentée par

$$f(z) = \frac{1}{2i\pi} \int_C w \, \frac{F_w(z,w)}{F(z,w)} \, dw \tag{21}$$

où $F_w(z,w)$ est la dérivée partielle par rapport à w de $F(z,w)$ et C est une circonférence $|w| = r$ avec $r < R_2$.

Preuve

La fonction

$$F(0,w) = a_{0,1}w + a_{0,2}w^2 + \ldots + a_{0,n}w^n + \ldots$$

admet un zéro simple en $w = 0$. Il existe un nombre r_2 $0 < r_2 \leq R_2$ tel que $F(0,w) \neq 0 \quad 0 < |w| < r_2$.

Lorsque z tend vers 0, $F(z,w)$ converge uniformément vers $F(0,w)$; par conséquent, pour r fixé, $0 < r < r_2$, il existe r_1, $0 < r_1 < R_1$ tel que

$$|F(z,w) - F(0,w)| < |F(0,w)| \quad , \quad |z| \leq r_1, \ |w| < r \tag{22}$$

Par le théorème de Rouché, ceci implique que pour tout z, $|z| \leq r_1$, $F(z,w)$ et $F(0,w)$ ont le même nombre de zéros sur $|w| < r$. $F(z,w)$ admet dès lors un zéro simple, $f(z)$, donné par la formule

$$f(z) = \frac{1}{2i\pi} \int_C w \, \frac{F_w(z,w)}{F(z,w)} \, dw$$

où C est une circonférence de centre 0 et de rayon r. Cette intégrale existe : $F(0,w) \neq 0$ sur C et donc par (22), $F(z,w) \neq 0$ pour $|z| \leq r_1$.

D'autre part, l'intégrale

$$\frac{1}{2i\pi}\int_C w\,\frac{F(z,w)\,F_{z,w}(z,w) - F_z(z,w)\,F_w(z,w)}{(F(z,w))^2}\,dw$$

existe et vaut $f'(z)$. $f(z)$ est donc holomorphe. ∎

2.2.5. Fonctions algébriques.

Considérons la fonction $F(z,w)$ définie en (17).

Théorème 14.

Soit z_0 un point ordinaire de $F(z,w)$, et si w_0 est une racine du polynôme $F(z_0,w)$, alors il existe une seule fonction $w(z,z_0)$, holomorphe dans un disque $|z-z_0| < r_1$, telle que

$$w(z_0,z_0) = w_0$$

$$F(z,w(z,z_0)) = 0 \qquad |z-z_0| < r_1$$

Preuve

Développons la fonction $F(z,w)$ en série autour de (z_0,w_0) et

$$F(z-z_0,w-w_0) = \sum_{j=1}^{n}\sum_{k=1}^{m}\frac{1}{j!\,k!}F_{j,k}(z_0,w_0)\,(z-z_0)^j\,(w-w_0)^k$$

où $m = \max_j m_j$ et $F_{j,k}(z,w)$ désigne la dérivée partielle de $F(z,w)$ d'ordre j par rapport à z et d'ordre k par rapport à w. Par les hypothèses sur z_0, nous avons que $F_{0,0}(z_0,w_0) = 0$ et $F_{0,1}(z_0,w_0) \neq 0$. Nous pouvons donc appliquer le théorème de la fonction implicite. Nous obtenons donc une fonction unique, $w(z,z_0)$, holomorphe dans un voisinage de z_0 : $|z-z_0| < r_1$, vérifiant

$$w(z_0,z_0) = w_0$$

$$F(z,w(z,z_0)) = 0 \qquad |z - z_0| < r_1$$

∎

La fonction $w(z,z_0)$ peut donc s'écrire sous la forme

$$w(z,z_0) = \sum_{n=1}^{\infty} d_n (z-z_0)^n + z_0 \qquad \text{où } d_1 \neq 0$$

Théorème 15.

Si z_0 est une racine simple du polynôme en w, $F(z,w_0)$ et si w_0 est une racine de multiplicité k; $1 < k \leq n$, du polynôme en w, $F(z_0,w)$, alors en un voisinage de z_0, il existe une fonction $g(Z)$, holomorphe pour $|Z|$ assez petit, les k déterminations de $g((z-z_0)^{1/k})$ représentant les k solutions de l'équation (18) qui en z_0 coïncident avec w_0.

Preuve

Les hypothèses sur z_0 et w_0 se traduisent sur les dérivées partielles de $F(z,w)$ par les relations

$$F_{0,0}(z_0,w_0) = 0$$

$$F_{0,1}(z_0,w_0) = \ldots = F_{0,k-1}(z_0,w_0) = 0$$

$$F_{0,k}(z_0,w_0) \neq 0$$

$$F_{1,0}(z_0,w_0) \neq 0$$

En récrivant l'équation (18) autour de (z_0,w_0), nous obtenons

$$z-z_0 = C_{2,0}(z-z_0)^2 + C_{1,1}(z-z_0)(w-w_0) + \ldots + C_{0,k}(w-w_0)^k + \ldots$$

où

$$C_{i,j} = - F_{i,j}(z_0,w_0) \,/\, i!j!\, F_{0,1}(z_0,w_0).$$

Nous avons que $C_{0,k} \neq 0$ et aucun terme ne contient uniquement une puissance de $(w-w_0)$ inférieure à k. L'application du théorème de la fonction implicite nous fournit la fonction solution

$$z-z_0 = C_{0,k}(w-w_0)^k + \gamma_{k+1}(w-w_0)^{k+1} + \ldots \qquad (23)$$

Aucun terme en $(w-w_0)$ de puissance inférieure à k ne peut apparaître puisque la fonction implicite est représentée par la formule (21).

Par le théorème 12, nous pouvons maintenant connaître les solutions de l'équation (23) et donc (18). Les k déterminations de la fonction $g((z-z_0)^{1/k})$ sont les k fonctions $w_i(z)$, $(i=1,\ldots,n)$, qui en z_0 coïncident avec w_0. Plus précisément, nous obtenons que, pour une détermination de $(z-z_0)^{1/k}$, la série

$$\sum_{n=1}^{\infty} d_n (z-z_0)^{n/k} + z_0$$

représente une des k solutions de l'équation (18).

2.3. Sensibilité des racines aux variations des coefficients.

2.3.1. Nombre de conditionnement.

Considérons un problème (P) dont les données sont les nombres a_i $(i=1,\ldots,n)$ et dont les solutions sont les nombres x_j $(j=1,\ldots,m)$.

Nous dirons que le problème P est mal conditionné si de petites erreurs relatives sur les données entraînent des erreurs relatives importantes sur les solutions. Il est clair que des problèmes de ce type sont indésirables, car pour obtenir un nombre déterminé de chiffres significatifs de la solution exacte, il nous faut alors déterminer les données avec un nombre de chiffres exacts plus conséquent.

Les nombres $k_{j,i}$ définis par

$$k_{j,i} = \frac{\partial x_j}{\partial a_i} \qquad (j=1,\ldots,m \text{ et } i=1,\ldots,n),$$

où ∂a_i est une erreur absolue sur la donnée a_i et ∂x_j est l'erreur absolue sur le résultat x_j qui en découle, nous donnent l'information nécessaire à la connaissance de la sensibilité des résultats aux variations des données.

Lorsque nous travaillons avec des nombres flottants, nous sommes surtout intéressés par les erreurs relatives sur les solutions. Donc, les nombres $K_{i,j}$ définis par la relation

$$\left|\frac{\partial x_j}{x_j}\right| \leq K_{j,i} \left|\frac{\partial a_i}{a_i}\right| \qquad (j=1,\ldots,m \text{ et } i=1,\ldots,n)$$

nous donnent l'information nécessaire. Nous appellerons ces quantités nombres de conditionnement du problème.

2.3.2. Sensibilité des racines aux variations des coefficients.

Considérons un polynôme défini par ses coefficients

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 \qquad (24)$$

dont les racines sont $z_1, z_2, \ldots, z_n$.

Considérons les racines du polynôme obtenu après perturbation des coefficients, c'est-à-dire les racines $z_r(\mathcal{E})$, $(r=1,\ldots,n)$, de $f(x) = p(x) + \mathcal{E}\, g(x)$ où $g(x) = b_n x^n + b_{n-1} x^{n-1} + \ldots + b_0$ est appelé polynôme perturbateur.

1) Cas d'une racine simple.

Supposons que z_r, $(r=1,\ldots,n)$, soit une racine simple de $p(x)$. Par le théorème 14, nous savons que la racine $z_r(\mathcal{E})$ de $f(x)$ est une fonction holomorphe en $\mathcal{E}$,

$$z_r(\mathcal{E}) = z_r + \sum_{k=1}^{n} P_k \mathcal{E}^k \qquad (25)$$

où la série en $\mathcal{E}$ est convergente pour $|\mathcal{E}|$ assez petit, et pour ces mêmes valeurs de $\mathcal{E}$, nous avons

$$\sum_{i=0}^{n} a_i (z_r+h)^i + \mathcal{E} \sum_{i=0}^{n} b_i (z_r+h)^i = 0$$

où $h = \sum_{k=1}^{n} P_k \mathcal{E}^k$.

Cette dernière relation peut encore s'écrire sous la forme

$$\sum_{i=0}^{n} c_i h^i + \mathcal{E} \sum_{i=0}^{n} d_i h^i = 0 \qquad (26)$$

où $c_i = \frac{1}{i!} p^{(i)}(z_r)$ et $d_i = \frac{1}{i!} g^{(i)}(z_r)$

Comme la formule (26) est vérifiée pour $|\mathcal{E}|$ assez petit, le coefficient de $\mathcal{E}$ (en particulier) doit s'annuler :

$$c_1 P_1 + d_0 = 0.$$

En faisant les identifications nécessaires, nous obtenons

$$P_1 = \frac{-g(z_r)}{p'(z_r)}$$

et nous pouvons écrire, par (25)

$$z_r(\varepsilon) = z_r - \varepsilon\, g(z_r) / p'(z_r) + O(\varepsilon^2) \qquad (27)$$

Si $g(x) = a_s x^s$, $(s=0,\ldots,n)$, alors

$$f(x) = a_n x^n + \ldots + a_s(1+\varepsilon)x^s + \ldots + a_0$$

et ε représente une erreur relative sur le coefficient a_s.

Les nombres de conditionnement $K_{r,s}$ valent donc :

$$K_{r,s} = \frac{-a_s z_r^{s-1}}{p'(z_r)} \qquad s=0,\ldots,n.$$

2) Cas d'une racine multiple.

Supposons maintenant que z_r, $(r=1,\ldots,n)$, soit une racine du polynôme $p(x)$, de multiplicité k, $(1 < k \leq n)$. Par le théorème 15, nous savons que

$$z_r(\varepsilon) = z_r + \sum_{j=1}^{\infty} P_j\, \varepsilon^{j/k}$$

représente une des k racines qui en $\varepsilon = 0$, coïncident avec z_r. La série en ε est convergente pour $|\varepsilon|$ assez petit et pour ces mêmes valeurs de ε

$$\sum_{i=0}^{n} a_i (z_r + h)^i + \varepsilon \sum_{i=0}^{n} b_i (z_r + h)^i = 0$$

où $h = \sum_{j=1}^{\infty} P_j\, \varepsilon^{j/k}$

Puisque $p(z_r) = p'(z_r) = \ldots = p^{(k-1)}(z_r) = 0$,

$p^{(k)}(z_r) \neq 0$.

L'égalité ci-avant, après des transformations similaires au cas précédent, peut s'écrire

$$\sum_{i=k}^{n} c_i h^i + \varepsilon \sum_{i=0}^{n} d_i h^i = 0$$

Puisque cette expression s'annule de manière identique au voisinage de $\varepsilon = 0$, le coefficent de $\varepsilon^{1/k}$ (en particulier) s'annule :

$$c_k \, P_1^{\,k} + d_0 = 0$$

Nous obtenons donc

$$P_1^{\,k} = \frac{-k!\; g(z_r)}{p^{(k)}(z_r)}$$

et, par conséquent,

$$z_r(\varepsilon) = z_r + \sqrt[k]{\frac{-k!\; g(z_r)}{p^{(k)}(z_r)}} \; \varepsilon^{1/k} + O(\varepsilon^{2/k})$$

Remarquons que si ε est une erreur relative de l'ordre de 10^{-16} sur un coefficient, alors l'erreur relative sur la racine z_r sera de l'ordre de $10^{-16/k}$. Désignons par ε' cette erreur relative :

$k = 2$ $\qquad \varepsilon' \sim 10^{-8}$

$k = 3$ $\qquad \varepsilon' \sim 10^{-5}$ $\qquad$ (*)

ce qui signifie que, dans les cas envisagés, tout au plus les 5 ou 8 premiers chiffres significatifs de la racine calculée seront fiables. La racine z_r est dite mal conditionnée.

(*) L'ordinateur Siemens 4004 travaille en chiffre hexadécimaux et dispose en double précision de 14 chiffres pour la mantisse, ce qui en base 10 revient à disposer d'environ 16 chiffres.

2.3.3. Quelques distributions typiques de racines.

Puisque les racines multiples sont mal conditionnées, nous pourrions nous attendre à ce que des racines proches l'une de l'autre soient mal conditionnées. D'autre part, des racines qui ne sont cependant pas proches peuvent néanmoins être mal conditionnées. Nous avons vu que pour des racines simples

$$z_r(\varepsilon) - z_r \sim -\varepsilon \frac{g(z_r)}{p'(z_r)} ,$$

mais le coefficient de ε peut devenir très important.

Considérons maintenant quelques distributions typiques de racines.

1) Distribution linéaire de racines.

Soit la distribution de racines : 1, 2, ..., 20. Pour considérer l'effet d'une perturbation du $k^{ème}$ coefficient sur les racines, prenons le polynôme perturbateur x^k

$$\partial z_r \sim \varepsilon \frac{z_r^k}{p'(z_r)} \qquad \begin{matrix} k=0,\ldots,20 \\ r=1,\ldots,20 \end{matrix}$$

$$\sim \varepsilon \frac{r^k}{(20-r)!\,(r-1)!} = \varepsilon A$$

A prend sa valeur maximale pour k = 19 et r = 16 : $A = 0.24\ 10^{10}$, alors que pour r=1, $A = 0.82\ 10^{-17}$.

Dans le cas où nous sommes intéressés par une perturbation relative du coefficient a_k,

$$\partial z_r \sim \varepsilon \frac{a_k r^k}{(20-r)!\,(r-1)!} = \varepsilon A$$

A prend alors sa valeur maximale pour r = 16, k = 15 : $A = 3.7\ 10^{14}$

Wilkinson montre ainsi que les racines proches de l'origine sont pratiquement insensibles aux variations des coefficients, tandis que les racines éloignées de l'origine sont d'autant plus sensibles.

Si nous considérons la distribution linéaire de racines k+1, ..., k+20, la sensibilité des racines devient plus importante si k croît. Si k=-10, alors le polynôme est mieux "conditionné" et la racine la plus sensible est la dix-huitième.

A prend sa valeur pour r = 18 et k = 19 :
$A = 0.20\ 10^{3}$.

2) Distribution géométrique de racines.

Comme second exemple, nous allons considérer un polynôme dont les racines sont $2^{-1}, 2^{-2}, \ldots, 2^{-20}$.

Supposons le coefficient a_{20} du polynôme normalisé à l'unité, nous avons les coefficients du polynômes p(x) qui obéissent à la majoration

$$|a_k| < 4.2^{-(1/2)(20-k)(21-k)} \qquad k=0,\ldots,20.$$

Nous avons

$$\partial z_r \sim -\partial a_k\ 2^{-kr} / P\ Q$$

où $P = (2^{-r} - 2^{-1})(2^{-r} - 2^{-2})\ldots(2^{-r} - 2^{-r+1})$

$Q = (2^{-r} - 2^{-r-1})(2^{-r} - 2^{-r-2})\ldots(2^{-r} - 2^{-20})$

Nous avons

$$P = (-1)^{r-1}\left[(1 - 2^{-r+1})\ldots(1 - 2^{-1})\right] / 2^{(1/2)\,r(r-1)}$$

$$Q = \left[(1 - 2^{-1})\ldots(1 - 2^{-20+r})\right] / 2^{r(20-r)}$$

Les expressions entre crochets convergent vers le produit infini

$$(1 - 2^{-1})(1 - 2^{-2})(1 - 2^{-3})\ \ldots$$

et une estimation assez grossière nous donne que ces expressions sont situées entre $\frac{1}{2}$ et $\frac{1}{4}$, de sorte que

$$\frac{1}{4}\, 2^{-\frac{1}{2}r(39-r)} > |PQ| > \frac{1}{16}\, 2^{-\frac{1}{2}r(39-r)}$$

et

$$|\partial z_r| < 16 . |\partial a_k| . 2^{(1/2)\, r(39-r-2k)}$$

A cause de la grande diversité en grandeur des racines, considérons les changements relatifs

$$\left|\frac{\partial z_r}{z_r}\right| < 16 \;\; |\partial a_k| \;\; 2^{(1/2)\, r(41-r-2k)}$$

Pour une valeur fixée de k, cette expression prend sa valeur maximale pour r = 20-k, de telle sorte que

$$\left|\frac{\partial z_r}{z_r}\right| < 16 . |\partial a_k| . 2^{(1/2)\, (20-k)(21-k)}$$

pour tout k.

Par la majoration constatée sur les coefficients a_k, nous obtenons finalement

$$\left|\frac{\partial z_r}{z_r}\right| < 64 \cdot \left|\frac{\partial a_k}{a_k}\right| \cdot |a_k| \;\; 2^{(1/2)\, (20-k)(21-k)}$$

$$< 64 \;\; \left|\frac{\partial a_k}{a_k}\right|$$

Cette relation montre que de petites erreurs relatives sur les coefficients ne peuvent provoquer des erreurs relatives importantes sur les racines. Ceci montre que pour des racines proches l'une de l'autre, leur rapport joue un rôle très important.

Wilkinson montre ensuite l'effet d'une perturbation sur le coefficient a_{19} du polynôme, égale à 2^{-31} sur les racines : avec 9 chiffres décimaux significatifs, seuls 3 des 20 racines sont modifiées et ce sur le dernier chiffre significatif uniquement.

3) Conclusion.

Nous aurions pu croire que, puisque des racines multiples sont mal conditionnées, les racines proches l'une de l'autre le seraient également.

Par ce dernier exemple, nous voyons que ce n'est pas toujours le cas; le rapport entre ces racines joue un rôle important.

Remarquons également que la grandeur des racines joue un rôle important. Plus les racines sont grandes, plus elles sont mal conditionnées.

2.4. Analyses d'erreur.

2.4.1. Les erreurs d'arrondi.

Si un programme s'effectue en nombres flottants, disposant de p chiffres, il existe une limite à la précision que l'on peut obtenir à chaque opération.

Considérons une opération comme étant la détermination d'un nombre x, fonction d'un ensemble de paramètres a_i, (i=1,...,n), appelées données :

$$x = g(a_1,...,a_n) \qquad (29)$$

cette fonction ne comportant que des opérations arithmétiques élémentaires.

Même si les données sont connues de manière précise, elles peuvent ne pas pouvoir se représenter en nombres flottants de p chiffres; nous devons dès lors nous contenter d'une approximation à p chiffres, source d'un premier type d'erreur : l'erreur inhérente. Remarquons cependant que pour des problèmes de type physique, chimique,..., la précision des observations est souvent inférieure à celle offerte par l'ordinateur, de telle sorte que les erreurs d'observation sont de loin plus importantes que les erreurs inhérentes à l'ordinateur.

D'autre part, la fonction g définie en (29) comporte un certain nombre d'opérations arithmétiques élémentaires et des erreurs d'approximation sont commises à chaque étape : c'est la deuxième source d'erreurs.

2.4.2. Analyse progressive-régressive.

Pour estimer la validité des résultats obtenus, nous disposons de deux types d'analyses : l'analyse progressive et l'analyse régressive.

1) Analyse progressive.

Déterminons la solution x à partir des données a_i, (i=1,...,n), par la relation (29). La valeur calculée $\bar{x}$ est différente de la valeur exacte, ceci à cause des erreurs d'arrondi. Dans ce type d'analyse, nous cherchons une borne pour l'expression

$$|\bar{x} - g(a_1,\ldots,a_n)| \qquad (30)$$

2) Analyse régressive.

Nous montrerons que la valeur calculée $\bar{x}$ est la valeur exacte obtenue à partir de (29) en perturbant les données

$$\bar{x} = g(a_1 + \delta_1,\ a_2 + \delta_2,\ldots,\ a_n + \delta_n)$$

et nous donnerons des bornes pour ces perturbations. Dans cette analyse, l'erreur d'approximation est exprimée sous forme d'une erreur inhérente.

Nous pourrions objecter que l'analyse régressive est incomplète, puisque ce qui nous intéresse finalement, ce n'est pas la perturbation des données, mais bien la différence (30). Pour compléter cette analyse, il nous reste à donner des bornes pour les variations des solutions, suite aux perturbations des données.

Remarque : L'exposé ci-dessus a été adapté au cas de l'ordinateur Siemens 4004 et nous avons envisagé la simple et la double précision. A cet effet, rappelons que les calculs se font en base 16 et que nous disposons de 6 chiffres pour la simple précision et de 14 chiffres pour la double précision.

2.5. Calculs des coefficients.

Dans la méthode usuelle, nous allons calculer les coefficients. Si nous disposons d'une relation de récurrence, comme dans le cas des polynômes de Legendre, nous pouvons facilement déterminer ces coefficients. En général, cependant, nous devons les calculer à partir des racines des deux polynômes factorisés. Notre but est de donner le polynôme calculé sous la forme

$$d_1(1 + E_1)x^n + d_2(1 + E_2)^{n-1} +\ldots+ d_{n+1}(1 + E_{n+1})$$

Calculons tout d'abord les coefficients d'un des deux polynômes, soit

$$p(x) = k\prod_{r=1}^{n} (x-z_r)$$

Nous allons calculer successivement les coefficients du polynôme

$$k\prod_{r=1}^{i} (x - z_r).$$

Supposons être à la $i^{\text{ème}}$ étape :

$$(d_1^{i-1}x^{i-1} + d_2^{i-1}x^{i-2} +\ldots+ d_i^{i-1})\ (x-z_i)$$

$$= d_1^{i-1}x^i + d_2^{i-1}x^{i-1} +\ldots+ d_i^{i-1}x$$

$$- (z_i d_1^{i-1}x^{i-1} +\ldots+ z_i d_{i-1}^{i-1}x) - z_i d_i^{i-1}$$

Les coefficients subissent donc les transformations suivantes :

$$d_1^i = d_1^{i-1} = k$$

$$d_j^i = d_j^{i-1} - z_i d_{j-1}^{i-1} \qquad (j=2,\ldots,i-1)$$

$$d_{i+1}^i = d_i^{i-1} z_i$$

Sur ordinateur, en double précision, les opérations s'effectuent de la manière suivante : supposons que les coef-

ficients calculés sont

$$\bar{d}_j^{i-1} = d_j^{i-1}(1 + \varepsilon_j^{i-1}) \qquad (j=2,\ldots,i)$$

$$\bar{d}_j^{i} = \bar{d}_j^{i-1}(1 + c_j) - z_i \bar{d}_{j-1}^{i-1}(1 + d_j)(1 + e_j) \qquad (j=2,\ldots,i-1)$$

$$\bar{d}_j^{i} = d_j^{i-1}(1 + \varepsilon_j^{i-1})(1 + c_j) - z_i d_{j-1}^{i-1}(1 + \varepsilon_{j-1}^{i-1})(1 + d_j + e_j)$$

où le terme en $d_j e_j$ a été négligé. Par des approximations similaires, nous obtenons

$$\bar{d}_j^{i} = d_j^{i-1}(1 + \varepsilon_j^{i-1} + c_j) - z_i d_{j-1}^{i-1}(1 + \varepsilon_{j-1}^{i-1} + d_j + e_j)$$

$$= d_j^{i}(1 + \varepsilon_j^{i}).$$

Dans un cadre général, nous ne pouvons cependant pas donner des bornes significatives pour $1 + \varepsilon_j^{i}$.

Reposons le problème. Nous voulons calculer la somme de deux nombres d_1 et d_2, dont nous connaissons des approximations $\bar{d}_1 = d_1(1 + e_1')$, $\bar{d}_2 = d_2(1 + e_2')$ avec

$$(1 - 16^{-12})^{n_i} \leq 1 + e_i' \leq (1 + 16^{-12})^{n_i} \qquad (i=1,2)$$

n_1 et n_2 sont deux nombres entiers déterminés précédemment.

Soit $d = d_1 + d_2$ et $\bar{d}$ la valeur calculée de la somme

$$\bar{d} = d_1(1 + e_1) + d_2(1 + e_2)$$

où $(1 - 16^{-12})^{n_i+1} \leq 1 + e_i \leq (1 + 16^{-12})^{n_i+1} \qquad (i=1,2)$

Soit e défini par la relation $\bar{d} = d(1 + e)$.

Nous avons :

$$d_1(1 + e_1) + d_2(1 + e_2) = d(1 + e)$$

$$d_1e_1 + d_2e_2 = d_1e + d_2e$$

$$e = (d_1e_1 = d_2e_2) / (d_1 + d_2)$$

Soit $E = \max\{|e_1|, |e_2|\}$, alors

$$|e| \leq \frac{(|d_1| + |d_2|)\ E}{|d_1 + d_2|}$$

Premier cas : d_1 et d_2 sont de même signe.

Alors $|d_1| + |d_2| = |d_1 + d_2|$

et nous pouvons donner des bornes satisfaisantes pour e :

$$(1 = 16^{-12})^{n+1} \leq 1 + e \leq (1 + 16^{-12})^{n+1}$$

où $n = \max\{n_1, n_2\}$.

Deuxième cas : d_1 et d_2 sont de signes différents.

Sans perte de généralité, supposons que d_2 soit supérieur à d_1 en norme, alors :

$$|e| \leq \frac{|d_1| + |d_2|}{|d_2| - |d_1|}\ E = KE$$

Nous voyons qu'aucune borne satisfaisante ne peut être donnée pour |e| : le facteur multiplicatif K est toujours supérieur à l'unité et devient très important lorsque le dénominateur de l'expression ci-dessus est quasiment nul. Nous pourrions éventuellement rétorquer que la majoration

$$|d_1e_1 + d_2e_2| \leq |d_1|\ E + |d_2|\ E$$

est trop importante dans le cas où d_1 et d_2 sont de signe opposé; mais rien ne nous permet d'affirmer que e_1 et e_2 sont de signes opposés. En conséquence, nous devons effectuer cette majoration.

En pratique, le calcul des coefficients peut s'effectuer sans erreur si les racines des deux polynômes factorisés exigent peu de chiffres pour leur représentation. Dans les autres cas, cela signifiera une perte des un ou deux derniers chiffres des coefficients.

Remarque : Si nous utilisons la simple précision dans les calculs, l'exposé ci-dessus n'est pas modifié dans une voie essentielle et les mêmes conclusions sont de rigueur.

2.6. Effets des erreurs d'arrondi sur la méthode de Newton-Raphson.

2.6.1. Evaluation du polynôme.

Cette évaluation se fait de manière classique par le schéma de Hörner. Ce schéma évalue le polynôme

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_0 \qquad (31)$$

par la formule

$$(\ldots((a_n x + a_{n-1})x + a_{n-2})\ldots) + a_0$$

Les calculs s'effectuent dans l'ordre suivant :

$$s_0 = a_n$$
$$s_1 = s_0 x + a_{n-1}$$
$$s_n = s_{n-1} x + a_0$$

Quel résultat obtenons-nous sur ordinateur ?

1) En double précision.

Les opérations d'addition et de multiplication s'effectuent suivant les règles

$$z_1 \boxplus z_2 = z_1(1 + e_1) + z_2(1 + e_2)$$

$$z_1 \boxdot z_2 = z_1 \cdot z_2 \cdot (1 + e_3)$$

où $|e_i| \leq 16^{-12}$ $\qquad (i=1,2,3)$

Le polynôme calculé est de la forme

$$a_n(1 + E_n)x^n + a_{n-1}(1 + E_{n-1})x^{n-1} + \ldots + a_0(1 + E_0)$$

où $(1 - 16^{-12})^{2i+1} \leq 1 + E_i \leq (1 + 16^{-12})^{2i+1}$

$\qquad (i=0,\ldots,n)$

En effet, les différentes étapes s'effectuent comme suit :

$$\overline{s_0} = a_n$$
$$\bar{s}_i = \bar{s}_{i-1} x(1 + e_i)(1 + d_i) + a_{n-1}(1 + c_i) \qquad (i=1,\ldots,n)$$

$$\text{où} \quad (1 - 16^{-12}) \leq 1 + \begin{matrix} c_i \\ d_i \\ e_i \end{matrix} \leq 1 + 16^{-12} \qquad (i=1,\ldots,n)$$

Par conséquent,

$$1 + E_n = (1 + e_1)(1 + d_1)(1 + e_2)\ldots(1 + e_n)(1 + d_n)$$

$$1 + e_{n-i} = (1 + c_i)(1 + e_{i+1})\ldots(1 + d_n) \qquad (i=1,\ldots,n)$$

Ces quantités sont donc comprises entre les bornes

$$(1 - 16^{-12})^{2n+1} \leq (1 - 16^{-12})^{2n} \leq 1 + E_n$$

$$\leq (1 + 16^{-12})^{2n} \leq (1 + 16^{-12})^{2n+1}$$

$$(1 - 16^{-12})^{2i+1} \leq 1 + E_i \leq (1 - 16^{-12})^{2i+1} \qquad (i=0,\ldots,n-1)$$

2) En simple précision.

Nous obtenons un résultat similaire. La seule modification est apportée par le fait que

$$\bar{s}_i = (\bar{s}_{i-1} x (1 + e_i) + a_{n-i})\ (1 + c_i) \qquad (i=1,\ldots,n)$$

Nous pouvons reprendre la même démonstration en postulant que $c_i = d_i$. Par conséquent, nous obtenons que le polynôme calculé est

$$a_n(1 + E_n)x^n + a_{n-1}(1 + E_{n-1})x^{n-1} + \ldots + a_0(1 + E_0)$$

$$\text{où} \quad (1 - 16^{-5})^{2i+1} \leq 1 + E_i \leq (1 + 16^{-5})^{2i+1} \qquad (i=0,\ldots,n)$$

2.6.2. La méthode de Newton-Raphson.

Soit z_k, une racine pas trop mal conditionnée du polynôme (31) (le sens de cette hypothèses deviendra plus clair ultérieurement). Les approximations $z_k + h_r$ sont déterminées par les formules :

$$z_k + h_{r+1} = z_k + h_r - \frac{p(z_k + h_r)}{p'(z_k + h_r)}$$

$$= z_k + h_r - \frac{p(z_k) + h_r p'(z_k) + \frac{h_r^2}{2} p''(z_k) + \ldots}{p'(z_k)}$$

$$= z_k + \frac{1}{2} \frac{h_r^2 \; p''(z_k) + \ldots}{p'(z_k) + \ldots}$$

où les termes omis au numérateur et au dénominateur comprennent les puissances supérieures de h_r. Si z_k est une racine isolée, alors $p'(z_k) \neq 0$, et nous avons :

$$h_{r+1} \sim \frac{p''(z_k)}{2p'(z_k)} \; h_r^2 \qquad (32)$$

à proximité de la racine.

2.6.3. Effet des erreurs d'approximation sur la méthode de Newton-Raphson.

Supposons que nous nous situons en un point $z_k + h_0$ tel que effectivement, la méthode de Newton-Raphson converge vers la racine z_k.

Notons par :

$z_k + \bar{h}_r$: les approximations successives calculées;

$z_k + h_{r+1}$: l'approximation exacte obtenue par la méthode de Newton-Raphson, à partir de $z_k + \bar{h}_r$;

$\bar{P}_r$, $\bar{P}'_r$: les valeurs calculées du polynôme et de sa dérivée en $z_k + \bar{h}_r$;

P_r, P'_r : les valeurs exactes du polynôme et de sa dérivée en $z_k + \bar{h}_r$.

Nous avons les erreurs absolues $|\bar{P}_r - P_r|$ et $|\bar{P}'_r - P'_r|$ qui sont du même ordre. A proximité de z_k, $|P_r|$ devient relativement petit face à $|P'_r|$ si z_k n'est pas une racine multiple. Dès lors, l'erreur relative commise lors du

calcul de la dérivée est négligeable face à l'erreur relative du polynôme.

Calculons tout d'abord le polynôme : la valeur exacte de $p(z_k + \bar{h}_r)$ est la valeur exacte d'un polynôme dont les coefficients sont $a_i(1 + E_i)$ et dont les racines sont $z(E)$ où $E = (E_0, E_1, \ldots, E_n)$:

$$\bar{P}_r = a_n(1 + E_n) \prod_{i=1}^{n} (z_k + \bar{h}_r - z_i(E))$$

$$P_r = a_n \prod_{i=1}^{n} (z_k + \bar{h}_r - z_i)$$

$$\frac{\bar{P}_r}{P_r} = (1 + E_n) \frac{\bar{h}_r + z_k - z_k(E)}{\bar{h}_r} \prod_{\substack{i=1 \\ \neq r}}^{n} \frac{z_k + \bar{h}_r - z_i(E)}{z_k + \bar{h}_r - z_i}$$

Pour autant que z_k soit suffisamment isolé des autres racines, le produit final est proche de l'unité, et ceci pour tout $\bar{h}_r$. Par conséquent, $\bar{P}_r/P_r$ est proche de l'unité aussi longtemps que les $\bar{h}_r$ considérés sont plus grands que $|z_k - z_k(E)|$. Nous avons supposé que l'erreur relative de la dérivée était négligeable face à l'erreur relative du polynôme. Par conséquent, nous pouvons écrire :

$$\frac{\bar{P}_r}{\bar{P}'_r} = (1 + B_r) \frac{P_r}{P'_r}$$

où B_r est principalement déterminé par $\bar{P}_r/P_r$ et sera négligeable aussi longtemps que $|\bar{h}_r|$ n'approche pas $|z_k - z_k(E)|$ en grandeur.

Considérons l'ordre de convergence de la méthode. A partir de (32), nous savons que pour $|\bar{h}_r|$ assez petit, nous avons la majoration

$$|h_{r+1}| < A\, h_{r+1}^2$$

La méthode de Newton-Raphson convergeant rapidement, nous pouvons supposer que

$$A|\bar{h}_r| < 0.1$$

D'autre part,

$$z_k + \bar{h}_{r+1} = z_k + \bar{h}_r - \frac{\bar{P}_r}{\bar{P}'_r}$$

$$z_k + \bar{h}_{r+1} = z_k + \bar{h}_r - (1 + B_r)\frac{P_r}{P'_r}$$

$$\bar{h}_{r+1} = \bar{h}_r - \frac{P_r}{P'_r} - B_r \frac{P_r}{P'_r} = h_{r+1} - B_r (\bar{h}_r - h_{r+1})$$

Par conséquent, nous avons la majoration suivante :

$$\begin{aligned} |\bar{h}_{r+1}| &\leq (1 + |B_r|)\,|h_{r+1}| + |B_r|\,|\bar{h}_r| \\ &\quad (1 + |B_r|)\;A\,h_r^2 + |B_r|\,|\bar{h}_r| \\ &\quad \left[0.1\,(1 + |B_r|) + |B_r|\right]\;|\bar{h}_r| \end{aligned} \qquad (33)$$

Il est clair que les erreurs d'approximations peuvent finalement détruire la convergence quadratique. Mais aussi longtemps que $|B_r| < 0.1$ (par exemple), nous avons $|\bar{h}_{r+1}| < 0.21|\bar{h}_r|$ de telle sorte que la convergence se prolonge de manière satisfaisante puisqu'il s'agit d'une convergence linéaire.

A partir de (33), en se souvenant que B_r est principalement déterminé par l'erreur relative commise lors de l'évaluation du polynôme, nous pouvons dire que la convergence cesse lorsque l'erreur commise en calculant P_r est d'un ordre de grandeur comparable à sa vraie valeur, de telle sorte que la valeur calculée n'a plus aucun chiffre significatif correct. Or, cette erreur vaut

$$\sum_{i=0}^{n} a_i E_i z_k^i,$$

tandis que la vraie valeur vaut approximativement $p'(z_k)\bar{h}_r$.

La convergence s'arrêtera donc lorsque

$$\frac{\sum a_i E_i z_k^i}{p'(z_k)\,\bar{h}_r} = O(1)$$

2.6.4. Interprétation géométrique.

Considérons la famille de polynôme

$$a_n(1 + E_n)x^n + a_{n-1}(1 + E_{n-1})x^{n-1} + \ldots + a_0(1 + E_0)$$

où $(1 - 16^{p'})^{2i+1} \leq 1 + E_i \leq (1 + 16^{-p'})^{2i+1}$ $(i=0,\ldots,n)$

p' est un nombre dépendant du nombre des chiffres de la mantisse.

Considérons les racines $z_i(E)$ de la famille du polynôme. Par les fonctions algébriques, nous savons que ces racines sont situées dans une boule de centre z_i et dont le rayon ρ_i dépend des nombres de conditionnement.

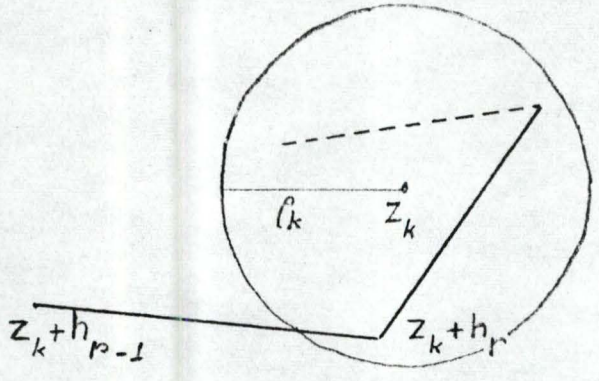


Fig. 2.1

Dès que h_r est d'un ordre de grandeur comparable à ρ_i, cette convergence quadratique de la méthode de Newton-Raphson se réduit à une convergence linéaire et, une fois arrivée dans la boule, le polynôme n'a plus aucun chiffre significatif exact. Il se produit un cyclage qui maintient le point courant $z_k + h_r$ dans cette boule.

Si la racine z_i n'est pas assez isolée, ou que les nombres de conditionnement sont trop importants, il peut

arriver que cette racine devienne de multiplicité supérieure à 1, ou même complexe. La figure 2.2 représente les cas suivants :

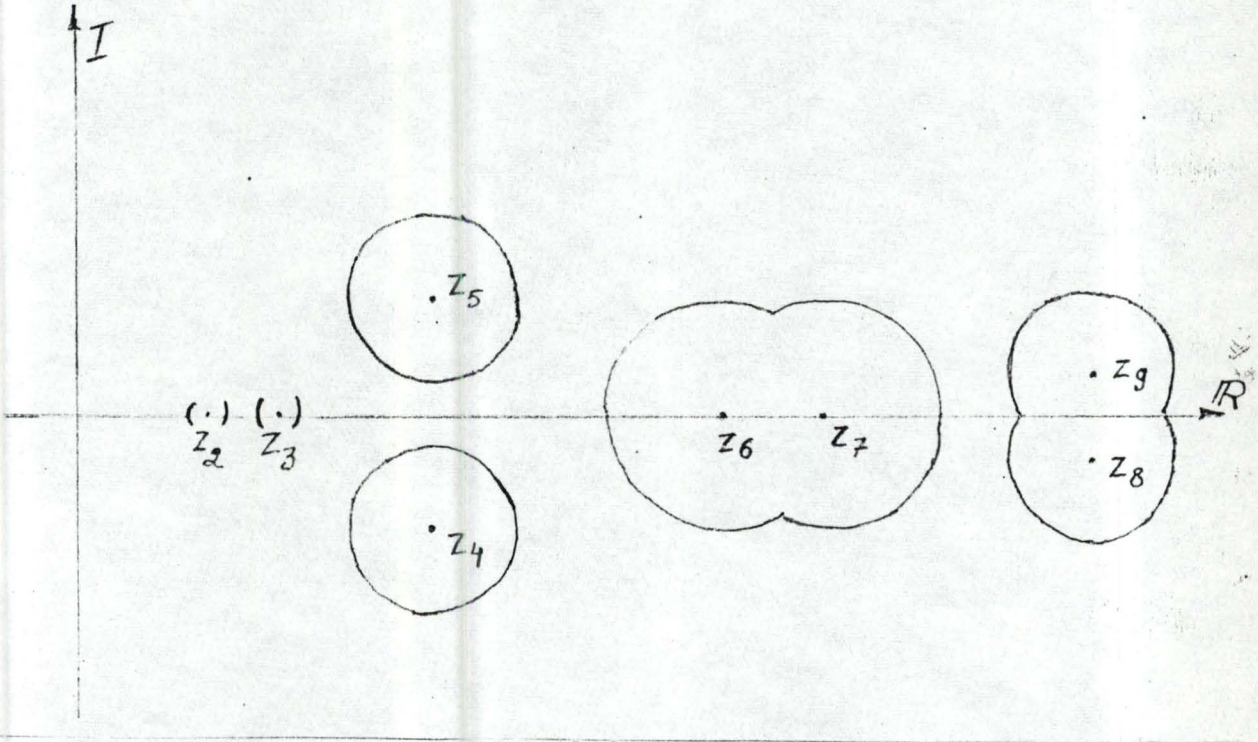


Fig. 2.2

z_2 et z_3 sont des racines proches qui restent réelles distinctes : elles sont bien conditionnées. z_4 et z_5 sont des racines complexes conjuguées qui le restent. z_8 et z_9 sont des racines complexes conjuguées qui peuvent devenir des racines doubles : elles sont mal conditionnées, tandis que z_6 et z_7 sont des racines distinctes réelles qui peuvent devenir complexes conjuguées.

3. LA METHODE PRODUIT.

3.1. Principe.

A partir des racines de deux polynômes factorisés, il y a naturellement moyen d'évaluer les polynômes et donc le polynôme somme sans passer par les coefficients. Il en est de même pour la dérivée de ce dernier polynôme. Par conséquent, la méthode de Newton-Raphson peut s'appliquer. Nous appellerons cette méthode, la méthode produit.

Resituons rapidement les notations

$$p_1(x) = k_1 \prod_{i=1}^{n} (x - a_i)$$

$$p_2(x) = k_2 \prod_{i=n}^{n} (x - b_i)$$

$$p(x) = p_1(x) + p_2(x)$$

3.2. Nombre de conditionnement du problème.

3.2.1. Sensibilité des racines aux variations des racines des polynômes factorisés.

Si $\mathcal{E}$ représente une erreur relative sur la racine a_i de $p_1(x)$, le polynôme sera de la forme

$$k_1(x - a_i)\dots(x - a_i - \mathcal{E}a_i)\dots(x - a_n) + p_2(x)$$
$$= p_1(x) + p_2(x) + (-\mathcal{E})a_i \frac{p_1(x)}{(x - a_i)}$$

Le polynôme perturbateur est donc

$$g(x) = a_i - \frac{p_1(x)}{(x - a_i)}$$

Puisque, dans le cas d'une racine simple

$$z_r(\mathcal{E}) = z_r + \mathcal{E}\frac{g(z_r)}{p'(z_r)} + O(\mathcal{E}^2)$$

$$\frac{\partial z_r}{z_r} = \frac{a_i p_1(z_r)}{z_r(z_r - a_i)p'(z_r)}$$

Les nombres de conditionnement sont donc

$$\overline{K}_{rs} = \frac{a_s p_1(z_r)}{z_r(z_r - a_s)p'(z_r)} \qquad (s=1,\dots,n)$$

De manière similaire, pour les racines du deuxième polynôme, nous obtenons :

$$\overline{K}_{rs} = \frac{b_s p_2(z_r)}{z_r(z_r - b_s)p'(z_r)} \qquad (s=1,\dots,m)$$

3.2.2. Sensibilité des racines aux variations des constantes multiplicatives k_1, k_2.

Si $\mathcal{E}$ représente une erreur relative sur la constante k_1, le polynôme sera de la forme

$$k_1(1 + \mathcal{E})(x - a_1)\dots(x - a_n) + p_2(x)$$
$$= p_1(x) + p_2(x) + \mathcal{E}p_1(x).$$

Le polynôme perturbateur est donc de la forme

$$g(x) = p_1(x)$$

et le nombre de conditionnement vaut

$$K_{r,n+m+1} = \frac{p_1(z_r)}{z_r\, p'(z_r)}$$

De manière similaire, si nous envisageons une erreur relative δ sur k_2, nous obtenons

$$K_{r,n+m+2} = \frac{p_2(z_r)}{z_r\, p'(z_r)}$$

Remarquons que l'un est l'opposé de l'autre.

3.3. Evaluation du polynôme.

Dans l'algorithme précédent, nous avons vu que les deux principales sources d'erreur étaient le calcul des coefficients et la méthode de Newton-Raphson. Dans ce deuxième algorithme, la seule source d'erreur provient de la méthode produit et, comme nous l'avons vu, les erreurs effectuées lors de l'évaluation du polynôme y occupe une place plus importante.

3.3.1. En simple précision.

Evaluons tout d'abord le premier polynôme. Les opérations s'effectuent dans l'ordre suivant :

$$t_0 = k_1$$

$$t_i = t_{i-1}(x - a_i) \qquad (i=1,\ldots,n)$$

Par conséquent, les valeurs calculées sont, lorsque nous travaillons en simple précision :

$$\bar{t}_0 = k_1$$

$$\bar{t}_i = \bar{t}_{i-1}(x - a_i)(1 + e_i)(1 + c_i)$$

où $(1 - 16^{-5}) \leq 1 + \begin{matrix} e_i \\ c_i \end{matrix} \leq 1 + 16^{-5} \qquad (i=1,\ldots,n)$

Par conséquent,

$$\bar{p}_1(x) = p_1(x)\ (1 + E_1')$$

où $1 + E_1' = (1 + e_1)(1 + c_1)\ldots(1 + e_n)(1 + c_n)$

Ce nombre obéit donc aux inégalités

$$(1 - 16^{-5})^{2n} \leq 1 + E_1' \leq (1 + 16^{-5})^{2n}$$

De manière similaire,

$$\bar{p}_2(x) = p_2(x)\ (1 + E_2')$$

$$(1 - 16^{-5})^{2m} \leq 1 + E_2' \leq (1 + 16^{-5})^{2m}$$

La valeur calculée du polynôme vaut donc

$$\bar{p}(x) = (1 + E_1')(1 + e)\, p_1(x) + (1 + E_2')(1 + e)\, p_2(x)$$

$$= (1 + E_1)\, p_1(x) + (1 + E_2)\, p_2(x)$$

où $(1 - 16^{-5})^{2n+1} \leq 1 + E_1 \leq (1 + 16^{-5})^{2n+1}$

$(1 - 16^{-5})^{2m+1} \leq 1 + E_2 \leq (1 + 16^{-5})^{2m+1}$

Conclusion : En simple précision, pour autant que les données soient représentables de manière exacte sur ordinateur, les perturbations enregistrées lors de l'évaluation du polynôme se transportent uniquement sur les constantes multiplicatives.

3.3.2. En double précision.

Considérons le premier polynôme :

$$\bar{t}_0 = k_1$$

$$\bar{t}_i = \bar{t}_{i-1}(x\,(1 + e_i) - a_i(1 + c_i))\,(1 + d_i)$$

$$= \bar{t}_{i-1}\left(x - a_i\frac{(1 + c_i)}{(1 + e_i)}\right)(1 + d_i)(1 + e_i)$$

$$= \bar{t}_{i-1}(x - a_i(1 + f_i))\,(1 + g_i)$$

où $(1 - 16^{-12})^2 \leq 1 + f_i \leq (1 + 16^{-12})^2$

Par conséquent,

$$\bar{p}_1(x) = (1 + E_1')k_1 \prod_{i=1}^{n} (x - a_i(1 + f_i))$$

où $(1 - 16^{-12})^{2n} \leq 1 + E_i' \leq (1 + 16^{-12})^{2n}$

De manière similaire,

$$\bar{p}_2(x) = (1 + E_2')k_2 \prod_{i=1}^{m} (x - b_i(1 + f_i'))$$

où $(1 - 16^{-12})^{2m} \leq 1 + E_2' \leq (1 + 16^{-12})^{2m}$

La valeur calculée du polynôme est donc

$$\overline{p}(x) = (1 + E_1)\, k_1 \prod_{i=1}^{n} (x - a_i(1 + f_i)) + (1 + E_2)\, k_2 \prod_{i=1}^{m} (x - b_i(1 + f_1'))$$

où $(1 - 16^{-12})^2 \leq 1 + \frac{f_i}{f_i'} \leq (1 + 16^{-12})^2$

$$(1 - 16^{-12})^{2n_i+1} \leq 1 + E_i \leq (1 + 16^{-12})^{2n_i+1} \qquad (i=1,2)$$

avec $n_1=n$, $n_2=m$.

Comparé à la simple précision, nous remarquons qu'en plus des perturbations sur les constantes multiplicatives, des perturbations apparaissent également au niveau des racines des deux polynômes factorisés.

4. COMPARAISON DES DEUX ALGORITHMES.

4.1. Description des programmes.

4.1.1. Premier algorithme : RERAC

Ce premier algorithme effectue la méthode dite usuelle. Elle comprend deux parties : le calcul des coefficients du polynôme $p(x)$ et la recherche de ses racines.

1) Calcul des coefficients.

La sous-routine COEFF calcule les coefficients d'un polynôme dont nous connaissons les racines. Il est appliqué aux deux polynômes

$$p_1(x) = k_1 \prod_{i=1}^{n} (x - a_i)$$

$$p_2(x) = k_2 \prod_{i=1}^{m} (x - b_i)$$

La méthode utilisée a été décrite dans ce présent chapitre (cfr section 2.2.5.). La sous-routine COPOLY calcule les coefficients du polynôme $p(x) = p_1(x) + p_2(x)$ à partir des coefficients des deux polynômes $p_1(x)$ et $p_2(x)$.

Ces résultats permettent de calculer le polynôme et sa dérivée en un point par un schéma de Hörner dans la sous-routine RESULT.

2) Recherche des racines.

Cette recherche se fait par la méthode de Newton-Raphson. La correction de Newton-Raphson est donc

$$- \frac{p(x)}{p'(x)}$$

Dans un premier temps, nous calculons la correction de Newton-Raphson relative à la fonction auxiliaire

$$f(x) = \frac{p(x)}{\prod_{i=1}^{n} (x - c_i)}$$

où les quantités c_i sont les n_c racines de p(x) déjà déterminées. La correction est alors de la forme

$$-\left(\frac{p'(x)}{p(x)} - \sum_{i=1}^{n_c} \frac{1}{x - c_i}\right)^{-1}$$

Lorsque nous arrivons au voisinage d'une racine non encore déterminée, nous considérons la correction de Newton-Raphson relative à la fonction initiale p(x). Ceci sera déterminé par un index N1 qui sera rendu non nul. Nous conviendrons que nous nous situons au voisinage d'une racine si nous vérifions la relation

$$| \text{CORX} / X | \leq 10^{-5}$$

CORX désigne la correction de Newton-Raphson évaluée au point X.

Le terme correctif $\sum_{i=1}^{n_c} \frac{1}{x - c_i}$ est calculé par la sous-routine SIRAC.

3) Critères de choix.

a) Correction de Newton-Raphson importante.

Ceci signifie que la factorisation est complète. La correction de Newton-Raphson est le rapport entre les valeurs du polynôme et de sa dérivée. Si la factorisation est complète, la fonction auxiliaire est une constante et la valeur de sa dérivée est donc nulle. Par les erreurs d'arrondi, il se peut cependant que cette correction ne soit pas infinie, mais importante. Si le cas se reproduit cinq fois au cours du processus d'une même racine, nous arrêterons donc le programme (au cours des itérations, nous pouvons parfois nous trouver au voisinage d'une racine de la dérivée et le même phénomène peut se reproduire). Un index N2 compte la répétition de ce cas.

b) Dans le cas contraire.

L'index N3 compte le nombre d'itérations de la méthode. Si au bout de cinquante itérations, nous ne sommes pas au voisinage d'une nouvelle racine (comme convenu ci-dessus), nous modifions le point de départ en ayant soin de lui donner une partie réelle et une partie imaginaire non nulle. En effet, dans le cas de polynômes à coefficients réels, si le point de départ de la méthode est un réel pur, nous resterons sur l'axe réel. Pour déterminer les racines complexes conjuguées, nous devons donc partir d'un point dont la partie imaginaire est non nulle. Arrivé au voisinage d'une racine, nous nous accordons dix itérations supplémentaires et le point obtenu sera considéré comme nouvelle racine du polynôme (l'index N2 comptabilise ces itérations), à moins que la relation

$$| \text{CORX} / X | \leq 5.10^{-16}$$

soit vérifiée, cas dans lequel le point X sera considéré comme racine. La recherche d'une racine se fait à l'intérieur de la sous-routine RAC.

La nature des résultats obtenus par cette sous-routine est renseignée par les valeurs de la variable K :

1) K = 0 : la méthode de Newton-Raphson ne converge pas;

2) K = 1 : la factorisation du polynôme est complète;

3) K 1 : une racine est trouvée.

Le point de départ de la méthode de Newton-Raphson sera donné par la $(n_c+1)^{\text{ème}}$ racine du second polynôme factorisé. Ceci impose que ce polynôme soit toujours du degré le plus élevé.

4) Analyse de la sensibilité.

Nous calculons les nombres de conditionnement relatifs à une racine si cette racine est simple et non nulle. Ces nombres sont donnés par les formules :

$$K_{r,s} = \frac{coef_s \; c_r^{s-1}}{p'(c_r)}$$

où $coef_s$, $(s=1,\ldots,n_s+1)$ désignent les coefficients du polynôme $p(x)$

$$p(x) = \sum_{i=1}^{n_s+1} coef_i \; x^{n_s+1-i}$$

4.1.2. Deuxième algorithme : RACCOM.

Ce deuxième algorithme effectue la méthode produit. Elle ne diffère de l'algorithme précédent que par la manière dont se calculent le polynôme et sa dérivée en un point.

1) Evaluation du polynôme (sous-routine RESULT).

Nous employons la formule :

$$p(x) = k_i \prod_{i=1}^{n} (x - a_i) + k_2 \prod_{i=1}^{m} (x - b_i)$$

2) Evaluation de la dérivée (sous-routine DERIV).

Nous calculons les dérivées $p_1'(x)$ et $p_2'(x)$ au point x et nous effectuons leur somme pour obtenir $p'(x)$.

Pour calculer la dérivée $p_1'(x)$ par exemple, nous procédons de la manière suivante :

a) X est situé à proximité d'une racine de $p_1(x)$:

Soit a_r cette racine.

$$p_1'(x) = \prod_{\substack{i=1 \\ \neq r}}^{n} (x - a_i)$$

b) Dans le cas contraire :

$$p_1'(x) = \sum_{i=1}^{n} \frac{p(x)}{x - a_i}$$

cette dernière expression étant calculée par la sous-routine SIRAC, identique à l'algorithme précédent.

3) Recherche des racines.

Elle s'effectue de manière similaire à l'algorithme précédent, dans une sous-routine RAC.

4) Analyse de la sensibilité.

Comme dans l'algorithme précédent, ne sont calculés que les nombres de conditionnement relatifs aux racines simples non nulles du polynôme. Ces nombres sont donnés par les formules :

$$K_{r,0} = \frac{p_1(c_r)}{c_r\, p'(c_r)}$$

$$\overline{K}_{r,i} = \frac{a_i\, p_1(c_r)}{c_r(z_r - a_i)\, p'(c_r)} \qquad (i=1,\ldots,n)$$

$$\overline{K}_{r,n+i} = \frac{b_i\, p_2(c_r)}{c_r(c_r - b_i)\, p'(c_r)} \qquad (i=1,\ldots,m)$$

4.2. Discussion des résultats numériques.

4.2.1. Préliminaires.

La théorie exposée à propos des effets des erreurs d'approximation sur la méthode de Newton-Raphson nous permet d'affirmer que, lorsque la convergence de la méthode est arrêtée, les chiffres des points courants, invariants pendant les dernières itérations, sont alors fiables et nous donnent une valeur approchée de la racine du polynôme. Dans la présentation des résultats, nous soulignerons les derniers chiffres que la méthode de Newton-Raphson a modifiés lors des précédentes itérations si la racine calculée n'est pas obtenue avec une précision maximale, c'est-à-dire si la convergence de la méthode est arrêtée. Les chiffres soulignés sont des chiffres non significatifs.

D'autre part, tout comme le fait Wilkinson dans la discussion de ses exemples, nous ne retiendrons comme nombre de conditionnement relatif à une même racine que le plus grand en valeur absolue. Il est significatif de la sensibilité de la racine aux variations des données. En effet, lors de l'évaluation du polynôme, toutes les données sont perturbées en même temps de manière similaire.

4.2.2. Exemple 1.

1) Données.

```
DEGRE DE P(X):  10
----------------

CONSTANTE MULTIPLICATIVE:
-------------------------
     ( 0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):
----------------
     ( 0.1000000000000000D 01, 0.0000000000000000D 00)
     ( 0.2000000000000000D 01, 0.0000000000000000D 00)
     ( 0.3000000000000000D 01, 0.0000000000000000D 00)
     ( 0.4000000000000000D 01, 0.0000000000000000D 00)
     ( 0.5000000000000000D 01, 0.0000000000000000D 00)
     ( 0.6000000000000000D 01, 0.0000000000000000D 00)
     ( 0.7000000000000000D 01, 0.0000000000000000D 00)
     ( 0.8000000000000000D 01, 0.0000000000000000D 00)
     ( 0.9000000000000000D 01, 0.0000000000000000D 00)
     ( 0.1000000000000000D 02, 0.0000000000000000D 00)


DEGRE DE P(X):  10
----------------

CONSTANTE MULTIPLICATIVE:
-------------------------
     ( 0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):
----------------
     ( 0.1500000000000000D 01, 0.0000000000000000D 00)
     ( 0.2500000000000000D 01, 0.0000000000000000D 00)
     ( 0.3500000000000000D 01, 0.0000000000000000D 00)
     ( 0.4500000000000000D 01, 0.0000000000000000D 00)
     ( 0.5500000000000000D 01, 0.0000000000000000D 00)
     ( 0.6500000000000000D 01, 0.0000000000000000D 00)
     ( 0.7500000000000000D 01, 0.0000000000000000D 00)
     ( 0.8500000000000000D 01, 0.0000000000000000D 00)
     ( 0.9500000000000000D 01, 0.0000000000000000D 00)
     ( 0.1050000000000000D 02, 0.0000000000000000D 00)
```

2) Résultats.

a) Par la méthode usuelle.

DEGRE DE PC(X): 10

RACINES DE PC(X):

(0.1414735743486591D 01, 0.0000000000000000D 00)
(0.2367608069575511D 01, 0.0000000000000000D 00)
(0.3330123327053399D 01, 0.0000000000000000D 00)
(0.4296803990340119D 01, 0.0000000000000000D 00)
(0.5265410585[illegible] 01, 0.0000000000000000D 00)
(0.6234589414400007D 01, 0.0000000000000000D 00)
(0.7203196009507626D 01, 0.0000000000000000D 00)
(0.8169876673134873D 01, 0.0000000000000000D 00)
(0.9132391930352065D 01, 0.0000000000000000D 00)
(0.1008526425653160D 02, 0.0000000000000000D 00)

CONSTANTE MULTIPLICATIVE:

(0.2000000000000000D 01, 0.0000000000000000D 00)

b) Par la méthode produit.

DEGRE DE PC(X): 10

RACINES DE PC(X):

(0.1414735743486715D 01, 0.0000000000000000D 00)
(0.2367608069575470D 01, 0.0000000000000000D 00)
(0.3330123327026700D 01, 0.0000000000000000D 00)
(0.4296803990322219D 01, 0.0000000000000000D 00)
(0.5265410585904339D 01, 0.0000000000000000D 00)
(0.6234589414095661D 01, 0.0000000000000000D 00)
(0.7203196009677781D 01, 0.0000000000000000D 00)
(0.8169876672973300D 01, 0.0000000000000000D 00)
(0.9132391930424530D 01, 0.0000000000000000D 00)
(0.1008526425651329D 02, 0.0000000000000000D 00)

CONSTANTE MULTIPLICATIVE:

(0.2000000000000004D 01, 0.0000000000000000D 00)

Considérons tout d'abord les racines obtenues par la méthode usuelle. Comme nous pouvions nous y attendre, ces racines sont assez mal conditionnées. Elles sont en effet très sensibles aux variations des coefficients. Les nombres de conditionnement des racines sont respectivement :

$0.1\ 10^3$	$0.25\ 10^4$	$0.25\ 10^5$	$0.12\ 10^6$
$0.43\ 10^6$	$0.86\ 10^6$	$0.11\ 10^7$	$0.96\ 10^6$
$0.42\ 10^6$	$0.86\ 10^5$		

Signalons la bonne correspondance entre l'ordre de grandeur de ces nombres et le nombre de chiffres non significatifs des racines.

Effectuons maintenant le changement de variable y = x-5. Théoriquement, comme le laisse prévoir Wilkinson, les racines doivent être mieux conditionnées. En effet, nous obtenons des résultats sans un seul chiffre significatif.

Résultat.

```
DEGRE DE PC(X):   10
------------------


RACINES DE PC(X):
-----------------
     (-0.3582642565132870 01, 0.0000000000000000D 00)
     (-0.2632391930424530D 01, 0.0000000000000000D 00)
     (-0.1669876572973300D 01, 0.0000000000000000D 00)
     (-0.7031960096777812D 00, 0.0000000000000000D 00)
     ( 0.2654105859043387D 00, 0.0000000000000000D 00)
     ( 0.1234569414095566D 01, 0.0000000000000000D 00)
     ( 0.2203196009677779D 01, 0.0000000000000000D 00)
     ( 0.3169876572973296D 01, 0.0000000000000000D 00)
     ( 0.4132391930424521D 01, 0.0000000000000000D 00)
     ( 0.5135264256513286D 01, 0.0000000000000000D 00)


CONSTANTE MULTIPLICATIVE:
-------------------------
     ( 0.2000000000000000D 01, 0.0000000000000000D 00)
```

Par contre, si nous effectuons le changement de variable y = x+k, avec k positif, les racines deviennent de plus en plus mal conditionnées. Pour k=20, nous n'obtenons même plus aucune racine.

Considérons maintenant les racines obtenues par la méthode produit. Elles sont toutes bien conditionnées et, si nous calculons les nombres de conditionnement, nous obtenons qu'ils sont tous de l'ordre de 10^{-1} au maximum. Pour vérifier les résultats obtenus, effectuons la translation : $y = x + 1.10^{-15}$. Nous obtenons

```
DEGRE DE PC(X):    10
---------------

RACINES DE PC(X):
-----------------
( 0.1414735743486715D 01, 0.0000000000000000D 00)
( 0.2367608069575471D 01, 0.0000000000000000D 00)
( 0.3330123327026701D 01, 0.0000000000000000D 00)
( 0.4296803990322220D 01, 0.0000000000000000D 00)
( 0.5265410583904340D 01, 0.0000000000000000D 00)
( 0.6234589414095662D 01, 0.0000000000000000D 00)
( 0.7203196009677782D 01, 0.0000000000000000D 00)
( 0.8169876674973301D 01, 0.0000000000000000D 00)
( 0.9132391930424331D 01, 0.0000000000000000D 00)
( 0.1008526426513290D 02, 0.0000000000000000D 00)

CONSTANTE MULTIPLICATIVE:
-------------------------
( 0.2000000000000003D 01, 0.6174374328226391D-15)
```

Les racines calculées ont également été soumises à la translation !. La précision est donc maximale.

D'autre part, nous avons :

$$\partial z_r \sim \frac{a_i \, p_1(z_r)}{(z_r - a_i)\, p'(z_r)} \varepsilon \quad = \quad A\,\varepsilon$$

où ε désigne une erreur relative sur la racine a_i.

Si nous effectuons une translation y=z+k, A se transforme de la manière suivante : A devient A + kA. Par conséquent, ce nombre ne tend plus vers 0 lorsque z_r s'approche de l'origine par translation. Théoriquement, cette racine devient donc très mal conditionnée. Vérifions le en effectuant la translation y = x - 5.26541.

Par la méthode usuelle, la cinquième racine calculée vaut

$$0.5859043387307878\ 10^{-6}$$

et il n'y a aucun nombre non significatif. Par la méthode produit, nous obtenons

$$0.5859043386898221\ 10^{-6}$$

Mais nous avons 7 chiffres non significatifs. Ceci est donc tout-à-fait compatible avec le nombre de conditionnement calculé.

Nous pouvons donc affirmer que les racines calculées par la méthode produit sont plus fiables que celles calculées par la méthode usuelle. Comparons maintenant les racines calculées dans les deux cas. Nous constatons que seuls les chiffres non significatifs différencient les résultats calculés.

Remarque : Vu la forme des racines des deux polynômes initiaux, les coefficients des polynômes seront calculés avec un minimum d'erreur.

4.2.3. Exemple 2.

Données.

a) Premier polynôme.

DEGRE DE P(X): 4

CONSTANTE MULTIPLICATIVE:

(0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):

(-1.7000000000000000D 01, 0.0000000000000000D 00)
(-0.5000000000000000D 01, 0.0000000000000000D 00)
(0.3000000000000000D 01, 0.0000000000000000D 00)
(0.1000000000000000D 02, 0.0000000000000000D 00)

b) Second polynôme.

DEGRE DE P(X): 4

CONSTANTE MULTIPLICATIVE:

(-0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):

(-0.6976[illegible] 01, 0.0000000000000000D 00)
(-0.5025[illegible] 01, 0.0000000000000000D 00)
(0.2996[illegible] 01, 0.0000000000000000D 00)
(0.1006[illegible] 02, 0.0000000000000000D 00)

Les racines du second polynôme ont été calculées avec une précision maximale, de manière à ce que la somme de ces deux polynômes se réduise à un polynôme de degré 1, dont la racine vaut l'unité. Par la méthode usuelle, si les coefficients du polynôme sont calculés de manière exacte, cette racine serait obtenue immédiatement avec une précision maximale. Par la méthode produit, nous obtenons :

0.9999999999997802

et le calcul du nombre de conditionnement maximum nous

donne une valeur 0.13 10^4. Cette racine est assez mal conditionnée.

4.2.4. Exemple 3.

Donnée.

```
DEGRE DE P(X):   4
-------------

CONSTANTE MULTIPLICATIVE:
------------------------
     ( 0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):
---------------
     (-0.7000000000000000D 01, 0.0000000000000000D 00)
     (-0.5000000000000000D 01, 0.0000000000000000D 00)
     ( 0.3000000000000000D 01, 0.0000000000000000D 00)
     ( 0.1000000000000000D 02, 0.0000000000000000D 00)



DEGRE DE P(X):   4
-------------

CONSTANTE MULTIPLICATIVE:
------------------------
     (-0.1000000000000000D 01, 0.0000000000000000D 00)

RACINES DE P(X):
---------------
     (-0.7175772[illegible]944916D 01, 0.0000000000000000D 00)
     (-0.4[illegible]4471D 01, 0.0000000000000000D 00)
     ( 0.2992905597[illegible]0D 01, 0.0000000000000000D 00)
     ( 0.1004528[illegible]5457360D 02, 0.0000000000000000D 00)
```

Nous devrions théoriquement obtenir une racine double valant l'unité, les racines du deuxième polynôme ayant été calculées de manière adéquate et obtenues avec une précision maximale. En fait, nous obtenons

1.000000878428469

0.9999998886564796

Nous perdons au moins la moitié des chiffres disponibles, comme nous l'avions d'ailleurs prévu théoriquement.

4.2.5. Exemple 4.

1) Données.

DEGRE DE P(X): 12

CONSTANTE MULTIPLICATIVE:

(0.106639381791400D-01, 0.000000000000000D 00)

RACINES DE P(X):

(0.000000000000000D 00,-0.195161724410200D 01)
(0.000000000000000D 00,-0.123329487104600D 01)
(0.000000000000000D 00,-0.107129817621200D 01)
(0.000000000000000D 00,-0.102416572658500D 01)
(0.000000000000000D 00,-0.100961368445700D 01)
(0.000000000000000D 00,-0.100541997873700D 01)
(0.000000000000000D 00, 0.100541997873700D 01)
(0.000000000000000D 00, 0.100961368445700D 01)
(0.000000000000000D 00, 0.102416572658500D 01)
(0.000000000000000D 00, 0.107129817621200D 01)
(0.000000000000000D 00, 0.123329487104600D 01)
(0.000000000000000D 00, 0.195161724410200D 01)

DEGRE DE P(X): 13

CONSTANTE MULTIPLICATIVE:

(0.100000000000000D 01, 0.000000000000000D 00)

RACINES DE P(X):

(0.000000000000000D 00,-0.999587256260400D 00)
(0.000000000000000D 00,-0.995430247699900D 00)
(0.000000000000000D 00,-0.981286498768900D 00)
(0.000000000000000D 00,-0.938114170560800D 00)
(0.000000000000000D 00,-0.814890277740200D 00)
(0.000000000000000D 00,-0.514957532298500D 00)
(0.000000000000000D 00, 0.000000000000000D 00)
(0.000000000000000D 00, 0.514957532298500D 00)
(0.000000000000000D 00, 0.814890277740200D 00)
(0.000000000000000D 00, 0.938114170560800D 00)
(0.000000000000000D 00, 0.981286498768900D 00)
(0.000000000000000D 00, 0.995430247699900D 00)
(0.000000000000000D 00, 0.999587256260400D 00)

2) Résultats.

a) Par la méthode usuelle.

```
DEGRE DE PC(X):    13
---------------

RACINES DE PC(X):
-----------------
     (-0.893744254494601100-03,-0.100016460560156880 01)
     ( 0.394856081122532301-02,-0.996822534111966300 00)
     (-0.125215530347405601-01,-0.985385310793670500 00)
     ( 0.377173915002741101-01,-0.949843692003748100 00)
     (-0.105264671283552670 00,-0.843307096314441100 00)
     ( 0.394856390269301110-02, 0.996822533424233700 00)
     (-0.125215530394414200-01, 0.985385309923206800 00)
     ( 0.377173915749173700-01, 0.949843691891776000 00)
     (-0.105264671283229840 00, 0.843307096311100300 00)
     (-0.893747363967006500-03, 0.100016460578641290 01)
     ( 0.238135817259413300 00, 0.555579172697522100 00)
     (-0.332907533896300290 00, 0.145254364848467900-15)
     ( 0.238135817259406400 00,-0.555579172697549400 00)

CONSTANTE MULTIPLICATIVE:
-------------------------
     ( 0.100000000000000000 01, 0.000000000000000000 00)
```

b) Par la méthode produit.

```
DEGRE DE PC(X):    13
---------------

RACINES DE PC(X):
-----------------
     (-0.893747363800033910-03,-0.100016460578820000 01)
     ( 0.394856387905139440-02,-0.996822533328567600 00)
     (-0.125215536703453210-01,-0.985385309943371400 00)
     ( 0.377173915710738810-01,-0.949843691891305000 00)
     (-0.105264671283396020 00,-0.843307096311054100 00)
     ( 0.394856387905155500-02, 0.996822533328567600 00)
     (-0.125215536703433660-01, 0.985385309943371400 00)
     ( 0.377173915710738810-01, 0.949843691891305000 00)
     (-0.105264671283396030 00, 0.843307096311054100 00)
     (-0.893747363800033200-03, 0.100016460573820590 01)
     ( 0.238135817259412100 00, 0.555579172697525100 00)
     (-0.332907533896300250 00, 0.241791788016939600-16)
     ( 0.238135817259412100 00,-0.555579172697525100 00)

CONSTANTE MULTIPLICATIVE:
-------------------------
     ( 0.100000000000000010 01,-0.410194519603079000-16)
```

Cet exemple est un cas comme on en rencontre souvent en synthèse des filtres; la plupart des commentaires effectués dans l'exemple 1 peuvent également trouver leur place ici. Remarquons en outre que, parmi les résultats calculés, il y a six paires de racines complexes qui sont conjuguées. Les racines obtenues par la méthode produit le montrent clairement, tandis que les racines obtenues par la méthode produit, en négligeant leurs chiffres non significatifs, ne sont pas encore complexes conjuguées. Il est nécessaire de supprimer encore deux chiffres supplémentaires. Ceci peut s'expliquer de la manière suivante. Vu la forme des racines des polynômes factorisés, les coefficients du polynôme somme vont se calculer avec des erreurs plus importantes qui vont essentiellement porter sur les un ou deux derniers chiffres de ces coefficients. Cette erreur se transporte sur les racines en rendant non significatifs un ou deux chiffres supplémentaires. Dans l'exemple 1, nous n'avons pas assisté à ce phénomène, vu la forme des racines des deux polynômes initiaux.

Remarquons finalement que les chiffres non significatifs sur la partie réelle et la partie imaginaire représentent des quantités de même ordre.

4.2.6. Conclusion.

La méthode produit nous fournira des bons résultats pour des problèmes où les racines des deux polynômes initiaux sont proches l'une de l'autre, de sorte que les racines du polynôme somme alternent avec les racines initiales et soient simples. Les quantités $p_1(x)$ et $p'(x)$ sont approximativement du même ordre pour les racines de $p(x)$. Une racine sera mal conditionnée si elle est proche de l'origine, comme le montre l'exemple 1.

Pour l'exemple 2, nous pouvons constater que les résultats ne sont pas toujours meilleurs par la méthode produit. Si le polynôme p(x) est de degré strictement inférieur aux degrés des polynômes initiaux, si la racine calculée se trouve isolée des racines données initialement, cette racine peut également devenir mal conditionnée.

B I B L I O G R A P H I E

1) WILKINSON, J.H. : Rounding Errors in Algebraic Processes. Prentice-Hall, Englewood Cliffs, N.J. (1963).

2) STERBENZ, P.H. : Floating-Point Computation. Prentice-Hall, Englewood Cliffs, N.J. (1974).

3) KNUTH, D. : The Art of Computer Programming. Vol 2. Addison Wesley (1969).

4) KULISCH, U. : An Axiomatic Approach to Rounded Computations. Num. Math., 18, 1-17 (1971-72).

5) KULISCH, U. : Formalization and Implementation of Floating - Point Arithmetic. Computing 14, 323-348 (1975).

6) KULISCH, U. : Rounding Invariant Structures. Mathematics Research Center. The University of Wisconsin, Madison, Wisconsin. Technical Summary Report Nr 1103, 1-47 (1970).

7) YOHE, M. : Roundings in Floating - Point Arithmetic. IEEE Transactions on Computers C22, 577-586 (1973).

8) Siemens System 4004. Reference Manual. Processor 201-219.

9) CHILOV, G. : Fonctions d'une variable. Tome 1, Editions de Moscou (1973).

10) THIRAN, J.P., VAN BASTELAER, Ph., WELLEKENS, C. : Description of a Filter Synthesis and Analysis Program with a Special Study of the Numerical Accuracy. Revue MBLE, Volume XIII, N° 2 (1966).

11) HILLE, E. : Analytic Function Theory. Vol 1. Chelsea Publishing Company New-York, N.Y. (1973).

TABLE DES MATIERES

AVANT-PROPOS . . . . . 1.

CHAP. I - Les opérations fondamentales sur l'ordinateur . . . . . 3.

1. Introduction . . . . . 3.

2. Représentation des nombres flottants sur ordinateur . . . . . 4.

2.1. Concept de grille . . . . . 4.

2.2. Introduction aux nombres flottants . . . . . 6.

2.3. Les "approximations" . . . . . 12.

2.4. Les "approximations" dans les ensembles totalement ordonnés . . . . . 17.

2.5. Les "approximations" symétriques . . . . . 20.

2.6. Etude de l'erreur . . . . . 22.

2.7. Représentation des nombres flottants sur l'ordinateur Siemens 4004 . . . . . 24.

3. L'arithmétique sur ordinateur . . . . . 26.

3.1. Point de vue axiomatique . . . . . 26.

3.2. Etude des opérations arithmétiques sur l'ordinateur . . . . . 30.

3.3. Addition et soustraction en virgule flottante . . . . . 32.

3.4. Multiplication en virgule flottante . . . . 44.

3.5. La division en virgule flottante . . . . . 50.

4. Les lois algébriques . . . . . 52.

4.1. Associativité de la loi additive . . . . . 52.

4.2. Associativité de la loi multiplicative . . . 54.

4.3. Distributivité de la loi multiplicative par rapport à la loi additive . . . . . 56.

4.4. Simplification à gauche et à droite . . . . 58.

4.5. Inverse pour la loi multiplicative . . . . . 59.

CHAP. II - Effet des erreurs d'arrondi sur un algorithme . . . . . 61.

1. Introduction . . . . . 61.

2. La méthode usuelle . . . . . 63.

2.1. Principe de la méthode . . . . . 63.

2.2. Les fonctions algébriques . . . . . 64.

2.3. Sensibilité des racines aux variations des coefficients . . . . . 73.

2.4. Analyses d'erreur . . . . . 81.

2.5. Calculs des coefficients . . . . . 83.

2.6. Effets des erreurs d'arrondi sur la méthode de Newton-Raphson . . . . . 87.

3. La méthode produit . . . . . 94.

3.1. Principe de la méthode . . . . . 94.

3.2. Nombre de conditionnement du problème . . . 95.

3.3. Evaluation du polynôme . . . . . 97.

4. Comparaison entre les deux algorithmes . . . . . 100.

4.1. Description des programmes . . . . . 100.

4.2. Discussion des résultats numériques . . . . 105.

BIBLIOGRAPHIE . . . . . 117.

TABLE DES MATIERES . . . . . 118.

```
      PROGRAM RERAC
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X,Y)
      DIMENSION A(20), B(20), C(20), COEF(20)
      READ 777, NP
  777 FORMAT (I2)
      DO 775 JP=1,NP


C     ***********************************************************
C     *                                                         *
C     *              TABLE DE VARIABLES                         *
C     *                                                         *
C     ***********************************************************
C
C     ***********************************************************
C     *                                                         *
C     *  DONNEES                                                *
C     *  *******                                                *
C     *        LES DEGRES DES POLYNOMES SONT DES ENTIERS LUS SOUS*
C     *  UN FORMAT (I2).                                        *
C     *        LES AUTRES DONNEES SONT DES NOMBRES COMPLEXES    *
C     *  EXPRIMES EN DOUBLE PRECISION. ILS SONT LUS SOUS UN FOR-*
C     *  MAT (2D25.18).                                         *
C     *        IL Y A UNE DONNEE PAR CARTE.                     *
C     *                                                         *
C     * NP     * NOMBRE DE PROBLEMES A TRAITER                 *
C     *        *                                               *
C     * NA     * DEGRE DU PREMIER POLYNOME                     *
C     * A(I)   * VECTEUR DES RACINES DU PREMIER POLYNOME       *
C     * CPA    * SA CONSTANTE MULTIPLICATIVE                   *
C     *        *                                               *
C     * NB     * DEGRE DU SECOND POLYNOME                      *
C     * CPB    * SA CONSTANTE MULTIPLICATIVE                   *
C     * B(I)   * VECTEUR DES RACINES DU SECOND POLYNOME        *
C     *        *                                               *
C     ***********************************************************
C     *                                                         *
C     *  RESULTATS                                              *
C     *  *********                                              *
C     *                                                         *
C     * NS     * DEGRE DU POLYNOME SOMME                       *
C     *COEF(I)* VECTEUR DES COEFFICIENTS DU POLYNOME SOMME     *
C     * C(I)   * LES RACINES DU POLYNOME SOMME                 *
C     * CPC    * CONSTANTE MULTIPLICATIVE DU POLYNOME SOMME    *
C     *        *                                               *
C     * DIF    * NOMBRE DE CONDITIONNEMENT DES RACINES PAR     *
C     *        * RAPPORT AUX DONNEES INTERMEDIAIRES QUE SONT LES *
C     *        * COEFFICIENTS                                  *
C     *        *                                               *
C     ***********************************************************
```

```
C      ********************************************************************
C      *                                                                  *
C      *  AUTRES VARIABLES                                                *
C      *  ****************                                                *
C      *                                                                  *
C      * NC     * NOMBRE DE RACINES DU POLYNOME SOMME DEJA                *
C      *        * CALCULEES                                               *
C      * M      * VERIFIE LA RELATION:  M = NC + 1                        *
C      *        * INDIQUE LA NATURE DES RESULTATS OBTENUS LORS DE         *
C      *        * LA RECHERCHE D'UNE RACINE                               *
C      * X      * POINT COURANT                                           *
C      * P      * VALEUR DU POLYNOME SOMME AU POINT X                     *
C      * D      * VALEUR DE SA DERIVEE AU POINT X                         *
C      *        *                                                         *
C      ********************************************************************
C
C
C
C
C      LECTURE ET IMPRESSION DES DONNEES
C      ---------------------------------
C
C      DANS LES DONNEES, LE DERNIER POLYNOME SERA TOUJOURS DE DEGRE LE
C       PLEVE.
C
       READ 1000, NA, CPA
       IF (NA.GT.0) READ 1001, (A(I),I=1,NA)
       READ 1000, NB, CPB
       IF (NB.GT.0) READ 1001, (B(I),I=1,NB)
 1000  FORMAT (I2,/,2D25.18)
 1001  FORMAT (2D25.18)
C
       PRINT 8999
 8999  FORMAT (1H1,5X,'DONNEES:',/,6X,8(1H*),//)
       PRINT 9000, NA, CPA
       IF (NA.GT.0) PRINT 9001,(A(I),I=1,NA)
       PRINT 9000, NB, CPB
       IF (NB.GT.0) PRINT 9001, (B(I),I=1,NB)
 9000  FORMAT (1HO,////,10X,'DEGRE DE P(X):',2X,I2,/,10X,13(1H-),
      1        //,10X,'CONSTANTE MULTIPLICATIVE:',/,10X,24(1H-),
      1        /,15X,'(',D23.16,',',D23.16,')')
 9001  FORMAT (1HO,9X,'RACINES DE P(X):',/,10X,15(1H-),/,
      1        (15X,'(',D23.16,',',D23.16,')'))
       PRINT 9002
 9002  FORMAT (1H1)
C
C
C      INITIALISATION
C      --------------
       CALL COPOLY (A,B,NA,NB,COEF,NS,CPA,CPB)
       IF (NS.EQ.0) GO TO 901
```

```
      NSP1 = NS + 1
      PRINT 9012, NS, (COEF(I), NSP1, I, I=1,NSP1)
 9012 FORMAT (1HO,///,10X,'DEGRE DE P(X):',I2,///,15X,
     1        'P(X)=   (',D23.16,',',D23.16,')* X**(',I2,'-',I2,')',/,
     1        (//,21X,'+ (',D23.16,',',D23.16,')* X**(',I2,'-',I2,')'))
      PRINT 9002
      NC = 0
      M = 1
   10 X = B(M)
C
C
C     RECHERCHE D'UNE RACINE.
C     -----------------------
      K = 0
      CALL RAC (X,COEF,NS,C,NC,K)
      IF ( K - 1 ) 100, 200,  300
C
C
C        LE PROCEDE NE CONVERGE PAS. ARRET DU PROGRAMME.
C        ***********************************************
  100 CPC = COEF(1)
      NS = NC
      GO TO 900
C
C
C        LA FACTORISATION EST COMPLETE.
C        ******************************
  200 CPC = COEF(1)
      NS = NC
      GO TO 901
C
C
C        UNE RACINE EST TROUVEE.
C        ***********************
  300 NC = NC + 1
      C(NC) = X
      IF (NC.EQ.NS) GO TO 200
C
C
C
C     NOUVEAU POINT DE DEPART. RECHERCHE D'UNE AUTRE RACINE.
C     ------------------------------------------------------
      M = NC + 1
      GO TO 10
C
C
C
C
C     IMPRESSION DES RESULTATS OBTENUS.
C     ---------------------------------
C
```

```
151   901    PRINT 9009
152   9009 FORMAT (1H1,20X,'LA FACTORISATION EST COMPLETE.',//,21X,
153      1        'LE DEGRE DU POLYNOME SOMME EST MAXIMUM.')
154   C
155   900   IF (NC.EQ.0) GO TO 911
156         PRINT 9010, NC, (C(I),I=1,NC)
157   9010 FORMAT (1H0,///,5X,'RACINES CALCULEES',/,5X,17(1H*),
158      1        ///,10X,'DEGRE DE PC(X):',3X,I2,/,10X,15(1H-),
159      1        ///,10X,'RACINES DE PC(X):',/,10X,17(1H-),/,
160      1        (15X,'(',D23.16,',',D23.16,')'))
161   911   PRINT 9011, CPC
162   9011 FORMAT (1H0,/,10X,'CONSTANTE MULTIPLICATIVE:',/,
163      1        10X,24(1H-),/,15X,'(',D23.16,',',D23.16,')')
164   C
165   C
166   C       ANALYSE DE LA SENSIBILITE DES RESULTATS
167   C       ---------------------------------------
168
169         PRINT 7000
170   7000 FORMAT (1H1,4X,'ANALYSE DE LA SENSIBILITE DES RESULTATS:',/,
171      1        5X,40(1H*),///)
172
173         DO 7100 K=1,NS
174         X = C(K)
175         PRINT 7050, K, X
176   7050 FORMAT (1H0,4X,'C(',I2,') = (',D23.16,',',D23.16,')')
177
178         D = CPC
179         DO 7060 J=1,NS
180         IF (J.EQ.K) GOTO 7060
181         D = D * (X - C(J))
182   7060 CONTINUE
183         D = D*X
184         IF ((DABS(DREAL(D)).LT.1.D-50).AND.
185      1        (DABS(DIMAG(D)).LT.1.D-50)) GO TO 7100
186
187         J = 0
188         DIF = COEF(NSP1)/D
189         PRINT 7070, K, J, DIF
190
191         PZ = (1.D0,0.D0)
192         DO 7080 J=1,NS
193         PZ = PZ * X
194         DIF = COEF(NSP1-J)*PZ/D
195         PRINT 7070, K, J, DIF
196   7070 FORMAT (1H0,4X,'DIF(',I2,',',I2,') = (',D23.16,',',D23.16')')
197   7080 CONTINUE
198         PRINT 7090
199   7090 FORMAT (1H0,//,65(1H=),//)
200   7100 CONTINUE
```

```
201           PRINT 9002
202
203
204    775    CONTINUE
205           STOP
206           END
```

```
      SUBROUTINE COPOLY (A,B,NA,NB,COEF,NS,CPA,CPB)
      IMPLICIT COMPLEX*16 (A-H,O-Z)
      DIMENSION A(1), B(1), AC(20), BC(20), COEF(20)
C     ***********************************************************
C     *                                                         *
C     *  VARIABLES D'ENTREE                                     *
C     *  ******************                                     *
C     *                                                         *
C     * NA     * DEGRE DU PREMIER POLYNOME                      *
C     * A(I)   * VECTEUR DES RACINES DU PREMIER POLYNOME        *
C     * CPA    * SA CONSTANTE MULTIPLICATIVE                    *
C     *        *                                                *
C     * NB     * DEGRE DU SECOND POLYNOME                       *
C     * B(I)   * VECTEUR DES RACINES DU SECOND POLYNOME         *
C     * CPB    * SA CONSTANTE MULTIPLICATIVE                    *
C     *        *                                                *
C     ***********************************************************
C     *                                                         *
C     *  VARIABLES DE SORTIE                                    *
C     *  *******************                                    *
C     *                                                         *
C     *COEF(I)* VECTEUR DES COEFFICIENTS DU POLYNOME SOMME      *
C     * NS     * DEGRE DU POLYNOME SOMME                        *
C     *        *                                                *
C     ***********************************************************
C     *                                                         *
C     *  AUTRES VARIABLES                                       *
C     *  ****************                                       *
C     *                                                         *
C     * AC(I) * COEFFICIENTS DU POLYNOME MONIQUE DE DEGRE NA    *
C     *       * DONT LES RACINES SONT A(I):I=1,...,NA           *
C     * BC(I) * COEFFICIENTS DU POLYNOME MONIQUE DE DEGRE NB    *
C     *       * DONT LES RACINES SONT B(I):I=1,...,NB           *
C     *       *                                                 *
C     ***********************************************************
      CALL COEFF (A,NA,AC)
      CALL COEFF (B,NB,BC)
      IF (NB.GT.NA)GOTO 205
```

```
51
52
53
54
55 C          1-ER CAS: NB = NA.
56 C          ******
57 C
58 C                CALCUL DES COEFFICIENTS.
59 C                ------------------------
60          NS = NA + 1
61          DO 210 I=1,NS
62   210    COEF(I) = CPA*AC(I) + CPB*BC(I)
63 C
64 C                CALCUL DU DEGRE DU POLYNOME SOMME.
65 C                ----------------------------------
66   212    IF (CDABS(COEF(1)).LT.1.D-50) GO TO 213
67          NS = NA
68          RETURN
69   213    DO 211 I=2,NS
70   211    COEF(I-1) = COEF(I)
71          NS = NS - 1
72          IF ( NS.EQ.1 ) GO TO 214
73          GOTO 212
74   214    PRINT 2001
75   2001   FORMAT (1H1,/////,5X,'LES DEUX POLYNOMES SE REDUISENT A UNE ',
76         1         'CONSTANTE.',/,5X,'ARRET DU PROGRAMME.')
77          NS = 0
78          RETURN
79 C
80 C
81 C          2-IEME CAS: NB > NA
82 C          **********
83 C       LE DEGRE DU POLYNOME SOMME VAUT NB.
84 C       IL FAUT SIMPLEMENT CALCULER LES COEFFICIENTS.
85 C
86   205    NBA=NB-NA
87          DO 206 I=1,NBA
88   206    COEF(I) = CPB*BC(I)
89          NBAP=NBA+1
90          NS = NB +1
91          DO 207 I=NBAP,NS
92   207    COEF(I) = CPA*AC(I-NBA) + CPB*BC(I)
93          NS = NB
94          RETURN
95          END
```

```
      SUBROUTINE COEFF (A,NA,AC)
      IMPLICIT COMPLEX*16 (A-H,O-Z)
      DIMENSION A(1), AC(1), C(20), E(20)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * NA     * DEGRE DU POLYNOME                                       *
C     * A(I)   * RACINES DE CE POLYNOME                                  *
C     *        *                                                         *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * AC(I) * COEFFICIENTS DE CE POLYNOME                              *
C     *       *                                                          *
C     ********************************************************************


      AC(1) = (1.D0,0.D0)
      DO 1 J=2,20
      AC(J) = (0.D0,0.D0)
    1 CONTINUE

      IF ( NA-1 ) 10, 20, 30

   10 RETURN

   20 AC(2) = -A(1)
      RETURN

   30 AC(2) = -A(1)
      DO 5 J=2,NA
      C(1) = (0.D0,0.D0)
      K=J+1
      DO 2 I=2,K
      C(I) = AC(I-1)*(-A(J))
    2 CONTINUE
      DO 4 I=1,K
      E(I) = C(I) + AC(I)
      AC(I) = E(I)
      C(I) = (0.D0,0.D0)
    4 CONTINUE
    5 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RAC (X,COEF,NS,C,NC,K)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DOUBLE PRECISION RCORX, ICORX
      DIMENSION C(1), COEF(1)
C
C
C
C     ******************************************************************
C     *                                                                *
C     *  VARIABLES D'ENTREE                                            *
C     *  ******************                                            *
C     *                                                                *
C     * X       * POINT DE DEPART DE LA METHODE DE NEWTON-RAPHSON      *
C     *COEF(I)* VECTEUR DES COEFFICIENTS DU POLYNOME SOMME             *
C     * NS      * DEGRE DU POLYNOME SOMME                              *
C     * NC      * NOMBRE DE RACINES DU POLYNOME SOMME DEJA             *
C     *         * CALCULEES                                            *
C     * C(I)    * LES RACINES EN QUESTION                              *
C     * K       * PERMET LES IMPRESSIONS INTERMEDIAIRES                *
C     *         *                                                      *
C     ******************************************************************
C     *                                                                *
C     *  VARIABLES DE SORTIE                                           *
C     *  *******************                                           *
C     *                                                                *
C     * X       * RACINE CALCULEE                                      *
C     * K       * INDIQUE LA NATURE DES RESULTATS OBTENUS LORS DE      *
C     *         * LA RECHERCHE D'UNE RACINE                            *
C     *         *                                                      *
C     ******************************************************************
C     *                                                                *
C     *  AUTRES VARIABLES                                              *
C     *  ****************                                              *
C     *                                                                *
C     * N1      * NOMBRE DE CORRECTIONS DE NEWTON-RAPHSON TROP         *
C     *         * IMPORTANTES                                          *
C     * N2      * NOMBRE D'ITERATIONS AU VOISINAGE D'UNE RACINE        *
C     * N3      * NOMBRE D'ITERATIONS                                  *
C     * X       * POINT COURANT                                        *
C     * P       * VALEUR DU POLYNOME SOMME AU POINT X                  *
C     * D       * VALEUR DE SA DERIVEE AU POINT X                      *
C     * CORX    * CORRECTION DE NEWTON-RAPHSON                         *
C     * CD      * TERME CORRECTIF                                      *
C     *         *                                                      *
C     ******************************************************************
C
C
```

```
C
C
C
C      INITIALISATION.
C      ---------------
       N1 = 0
       N2 = 0
       N3 = 0
       IF ( K.EQ. 0 ) PRINT 8
 8     FORMAT (1HO,////////,11X,'POINT COURANT:',39X,'CORRECTION:',//)
C
C
C      CALCUL DE LA CORRECTION DE NEWTON-RAPHSON.
C      ------------------------------------------
 1     CALL RESULT (COEF,NS,X,P,D)
       IF ((DABS(DREAL(P)).LT.1.D-50).AND.(DABS(DIMAG(P)).LT.1.D-50))
      1        GO TO 401
       IF ((DABS(DREAL(D)).LT.1.D-50).AND.(DABS(DIMAG(D)).LT.1.D-50))
      1        GO TO 99
       IF (N1.EQ.0) GO TO 50
       CORX = (-1.DO,0.DO) * P/D
       GO TO 100
C
 50    CALL SIRAC (X,C,NC,1.DO,CD)
       D = (D/P) - CD
       IF ((DABS(DREAL(D)).LT.1.D-50).AND.(DABS(DIMAG(D)).LT.1.D-50))
      1        GO TO 99
       CORX = (-1.DO,0.DO) / D
C
 100   RCORX = DREAL(CORX)
       ICORX = DIMAG(CORX)
       IF((DABS(ICORX).GT.1.D-20).AND.
      1       (DABS(RCORX/ICORX).LE.1.D-20)) RCORX=0.D-16
       IF ((DABS(RCORX).GT.1.D-20).AND.
      1       (DABS(ICORX/RCORX).LE.1.D-20)) ICORX=0.D-16
       CORX = DCMPLX(RCORX,ICORX)
C
C
C      CAS D'UNE CORRECTION DE N-R. IMPORTANTE.
C      ----------------------------------------
       IF ( CDABS(CORX).LE.1.D05) GO TO 200
 99    N2 = N2 + 1
       IF (N2.EQ.6) GO TO 101
       X = X + (0.1DO,0.1DO)
       N1 = 0
       GO TO 1

 101   K = 1
       RETURN
```

```
101 C
102 C
103 C      DANS LE CAS CONTRAIRE.
104 C      ----------------------
105  200   IF (K.EQ.0) PRINT 9, X, CORX
106  9     FORMAT ( 2(5X,'(',D23.16,',',D23.16,')'))
107        IF ((DABS(DREAL(X)).GT.1.D-50).OR.
108       1        (DABS(DIMAG(X)).GT.1.D-50)) GO TO 250
109        X = X + CORX
110        N3 = N3 + 1
111        IF (N3-50) 1, 300, 400
112 C
113 C      LA PRECISION OBTENUE EST INFERIEURE A 10**-5.
114 C      ---------------------------------------------
115  250   IF (CDABS(CORX/X).LT.1.D-5) GO TO 500
116        X = X + CORX
117        N3 = N3 + 1
118        IF ( N3 - 50 )  1, 300, 400
119  400   IF (N3.LT.150) GO TO 1
120        IF (K.EQ.0) PRINT 9100
121  9100  FORMAT (1HO,5X,'LE PROCESSUS NE CONVERGE PAS.')
122        K = 0
123        RETURN
124 C
125  300   X = (-1.D0,-1.D0)
126        N1 = 0
127        GO TO 1
128 C
129 C
130 C      LA PRECISION OBTENUE EST SUPERIEURE A 10**-5.
131 C      ---------------------------------------------
132 C         1ER CAS: LA PRECISION OBTENUE  EST MAXIMALE
133 C         ========
134  500   IF (CDABS(CORX/X).GT.5.D-16 ) GO TO 600
135        X = X + CORX
136        IF (K.EQ.0) PRINT 9300, X
137  9300  FORMAT (1HO,///,5X,'RACINES OBTENUEES AVEC UNE PRECISION ',
138       1        'MAXIMALE:',//,15X,'(',D23.16,',',D23.16,')')
139        K = 3
140        RETURN
141 C
142 C
143 C         2IEME CAS: LA PRECISION OBTENUEE N'EST PAS MAXIMALE.
144 C         ==========
145  600   N1 = N1 + 1
146        X = X + CORX
147        IF ( N1.LT.10 ) GO TO 1
148  401   IF (K.EQ.0) PRINT 9400, X
149  9400  FORMAT (1HO,///,5X,'RACINES TROUVEES SANS PRECISION MAXIMALE:',
150       1        //,15X,'(',D23.16,',',D23.16,')')
```

```
151         K = 4
152         RETURN
153         END
```

```
      SUBROUTINE RESULT(COEF,N,X,PX,DX)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DIMENSION COEF(1)


C     ****************************************************************
C     *                                                              *
C     *  VARIABLES D'ENTREE                                          *
C     *  ******************                                          *
C     *                                                              *
C     * N       * DEGRE DU POLYNOME                                  *
C     *COEF(I)* COEFFICIENTS DE CE POLYNOME                         *
C     * X       * POINT COURANT                                      *
C     *         *                                                    *
C     ****************************************************************
C     *                                                              *
C     *  VARIABLES DE SORTIE                                         *
C     *  *******************                                         *
C     *                                                              *
C     * PX      * VALEUR DU POLYNOME AU POINT X                      *
C     * DX      * VALEUR DE SA DERIVEE AU POINT X                    *
C     *         *                                                    *
C     ****************************************************************


      PX = (0.D0,0.D0)
      NP1 = N + 1
      DO 20 J=1,NP1
   20 PX = PX * X + COEF(J)
      DX = (0.D0,0.D0)
      DO 30 J=1,N
   30 DX = DX * X + (NP1 - J ) * COEF(J)
      RETURN
      END
```

```
      SUBROUTINE SIRAC(X,A,NA,PA,AD)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DIMENSION A(1)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * X       * POINT COURANT                                          *
C     * NA      * DEGRE DU POLYNOME                                      *
C     * A(I)    * RACINES DE CE POLYNOME                                 *
C     * PA      * VALEUR DU POLYNOME AU POINT X                          *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * AD      * VALEUR DE SA DERIVEE AU POINT X                        *
C     *         *                                                        *
C     ********************************************************************


      IF ( NA.GT.0 ) GO TO 10
      AD = (0.D0,0.D0)
      RETURN
C
   10 AD = (0.D0,0.D0)
      DO 20 I = 1, NA
      XI = X - A(I)
      IF ((DABS(DREAL(XI)).LT.1.D-50) .AND.
     1       (DABS(DIMAG(XI)).LT.1.D-50)) GO TO 30
      AD = AD + PA/XI
   20 CONTINUE
      RETURN

   30 AD = (0.D0,0.D0)
      RETURN
      END
```

```
      PROGRAM RACCOM
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X,Y)
      DIMENSION A(20), B(20), C(20)
      READ 777, NP
  777 FORMAT (I2)
      DO 775 JP=1,NP
C
C*****************************************************************
C*                                                               *
C*                    TABLE DE VARIABLES                         *
C*                                                               *
C*****************************************************************
C
C*****************************************************************
C*                                                               *
C*  DONNEES                                                      *
C*  *******                                                      *
C*        LES DEGRES DES POLYNOMES SONT DES ENTIERS LUS SOUS     *
C*  UN FORMAT (I2).                                              *
C*        LES AUTRES DONNEES SONT DES NOMBRES COMPLEXES          *
C*  EXPRIMES EN DOUBLE PRECISION. ILS SONT LUS SOUS UN FOR-      *
C*  MAT (2D25.18).                                               *
C*        IL Y A UNE DONNEE PAR CARTE.                           *
C*                                                               *
C*  NP      * NOMBRE DE PROBLEMES A TRAITER                      *
C*          *                                                    *
C*  NA      * DEGRE DU PREMIER POLYNOME                          *
C*  A(I)    * VECTEUR DES RACINES DU PREMIER POLYNOME            *
C*  CPA     * SA CONSTANTE MULTIPLICATIVE                        *
C*          *                                                    *
C*  NB      * DEGRE DU SECOND POLYNOME                           *
C*  B(I)    * VECTEUR DES RACINES DU SECOND POLYNOME             *
C*  CPB     * SA CONSTANTE MULTIPLICATIVE                        *
C*          *                                                    *
C*****************************************************************
C*                                                               *
C*  RESULTATS                                                    *
C*  *********                                                    *
C*                                                               *
C*  C(I)    * LES RACINES DU POLYNOME SOMME                      *
C*  CPC     * CONSTANTE MULTIPLICATIVE DU POLYNOME SOMME         *
C*  DIF     * NOMBRE DE CONDITIONNEMENT DU PROBLEME              *
C*          *                                                    *
C*****************************************************************
```

```
C     ********************************************************************
C     *                                                                  *
C     *  AUTRES VARIABLES                                                *
C     *  ****************                                                *
C     *                                                                  *
C     * X       * POINT COURANT                                          *
C     * P       * VALEUR DU POLYNOME SOMME AU POINT X                    *
C     * D       * VALEUR DE SA DERIVEE AU POINT X                        *
C     * PA      * VALEUR DU PREMIER POLYNOME AU POINT X                  *
C     * DA      * VALEUR DE SA DERIVEE AU POINT X                        *
C     * PB      * VALEUR DU SECOND POLYNOME AU POINT X                   *
C     * DB      * VALEUR DE SA DERIVEE AU POINT X                        *
C     * NC      * NOMBRE DE RACINES DU POLYNOME SOMME DEJA               *
C     *         * CALCULEES                                              *
C     * M       * VERIFIE LA RELATION:  M = NC + 1                       *
C     * PC      * VERIFIE LA RELATION:  P = CPC * PC                     *
C     * K       * PERMET LES IMPRESSIONS INTERMEDIAIRES                  *
C     *         * INDIQUE LA NATURE DES RESULTATS OBTENUS LORS DE         *
C     *         * LA RECHERCHE D'UNE RACINE                              *
C     *         *                                                        *
C     ********************************************************************
C
C
C
C
C     LECTURE ET IMPRESSION DES DONNEES
C     ---------------------------------
C
C     DANS LES DONNEES, LE DERNIER POLYNOME SERA TOUJOURS DE DEGRE LE P
C      ELEVE.
C
      READ 1000, NA, CPA
      IF (NA.GT.0) READ 1001, (A(I),I=1,NA)
      READ 1000, NB, CPB
      IF (NB.GT.0) READ 1001, (B(I),I=1,NB)
 1000 FORMAT (I2,/,2D25.18)
 1001 FORMAT (2D25.18)
C
      PRINT 8999
 8999 FORMAT (1H1,5X,'DONNEES:',/,6X,8(1H*),//)
      PRINT 9000, NA, CPA
      IF (NA.GT.0) PRINT 9001,(A(I),I=1,NA)
      PRINT 9000, NB, CPB
      IF (NB.GT.0) PRINT 9001, (B(I),I=1,NB)
 9000 FORMAT (1H0,////,10X,'DEGRE DE P(X):',2X,I2,/,10X,13(1H-),
     1       //,10X,'CONSTANTE MULTIPLICATIVE:',/,10X,24(1H-),
     1       /,15X,'(',D23.16,',',D23.16,')')
 9001 FORMAT (1H0,9X,'RACINES DE P(X):',/,10X,15(1H-),/,
     1       (15X,'(',D23.16,',',D23.16,')'))
      PRINT 9002
 9002 FORMAT (1H1)
```

```
C
C
C       INITIALISATION
C       --------------
        NC = 0
        M = 1
 10     X = B(M)
C
C
C       RECHERCHE D'UNE RACINE.
C       -----------------------
        K = 0
        CALL RAC (X,A,NA,CPA,B,NB,CPB,C,NC,K)
        IF ( K - 1 ) 100, 200,  300
C
C
C          LE PROCEDE NE CONVERGE PAS. ARRET DU PROGRAMME.
C          ***********************************************
 100    PRINT 9100
 9100   FORMAT (1H1,20X,'LE PROCESSUS NE CONVERGE PLUS.')
        GO TO 900
C
C
C          LA FACTORISATION EST COMPLETE.
C          ******************************
 200    PRINT 9200
 9200   FORMAT (1H1,20X,'LA FACTORISATION EST COMPLETE.',//,5X,
     1          'A REMARQUER QUE LE DEGRE DU POLYNOME SOMME',/,5X,
     1          'EST INFERIEUR AU DEGRE MAXIMUM NB.')
        GO TO 900
C
C
C          UNE RACINE EST TROUVEE.
C          ***********************
 300    NC = NC + 1
        C(NC) = X
        IF ( NC.EQ.NB ) GO TO 901
C
C
C       NOUVEAU POINT DE DEPART. RECHERCHE D'UNE AUTRE RACINE.
C       ------------------------------------------------------
        M = NC + 1
        GO TO 10
C
C
C
C       IMPRESSION DES RESULTATS OBTENUS.
C       ---------------------------------
 901    PRINT 9009
 9009   FORMAT (1H1,20X,'LA FACTORISATION EST COMPLETE.',//,21X,
```

```
151         1        'LE DEGRE DU POLYNOME SOMME EST MAXIMUM.')
152   C
153    900   IF (NC.EQ.0 ) GO TO 650
154         CALL RESULT ((5.D0,0.D0),A,NA,CPA,PA)
155         CALL RESULT ((5.D0,0.D0),B,NB,CPB,PB)
156         P = PA + PB
157         CPC = (1.D0,0.D0)
158         CALL RESULT ((5.D0,0.D0),C,NC,CPC,PC)
159         CPC = P / PC
160         PRINT 9010, NC, (C(I),I=1,NC)
161    9010 FORMAT (1H0,///,5X,'RACINES CALCULEES',/,5X,17(1H*),
162         1        ///,10X,'DEGRE DE PC(X):',3X,I2,/,10X,15(1H-),
163         1        ///,10X,'RACINES DE PC(X):',/,10X,17(1H-),/,
164         1        (15X,'(',D23.16,',',D23.16,')'))
165    650  PRINT 9011, CPC
166    9011 FORMAT (1H0,/,10X,'CONSTANTE MULTIPLICATIVE:',/,
167         1        10X,24(1H-),/,15X,'(',D23.16,',',D23.16,')')
168
169   C
170   C          ANALYSE DE LA SENSIBILITE DES RESULTATS
171   C          ---------------------------------------
172         PRINT 8005
173    8005 FORMAT (1H1,4X,'ANALYSE DE LA SENSIBILITE DES RESULTATS:',
174         1        /,5X,40(1H*),//)
175         DO 8003 K=1,NC
176         X = C(K)
177         CALL RESULT (X,A,NA,CPA,PA)
178         CALL DERIV(X,A,NA,CPA,PA,DA)
179         PB = -PA
180         CALL DERIV (X,B,NB,CPB,PB,DB)
181         D = DA + DB
182         IF ((DABS(DREAL(D)).LT.1.D-50).AND.
183         1        (DABS(DIMAG(D)).LT.1.D-50)) GO TO 8003
184         PRINT 8000, K, C(K)
185    8000 FORMAT (1H0,4X,'C(',I2,') = (',D23.16,',',D23.16,')')
186         D = (PA/X)/D
187         PRINT 8001, K, D
188    8001 FORMAT (1H0,4X,'DIF(',I2,', 0) = (',D23.16,',',D23.16,')')
189         DO 50 J=1,NA
190         DIF = (A(J)/(X-A(J)))*D
191    50   PRINT 8002, K, J, DIF
192         DO 51 J=1,NB
193         DIF = (B(J)/(X-B(J)))*D
194    51   PRINT 8002, K, J, DIF
195    8002 FORMAT (1H0,4X,'DIF(',I2,',',I2,') = (',D23.16,',',D23.16,')')
196    8003 PRINT 8004
197    8004 FORMAT (1H0,65(1H=),///)
198         PRINT 9002
199    775  CONTINUE
200         STOP
```

```
      SUBROUTINE RAC (X,A,NA,CPA,B,NB,CPB,C,NC,K)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DOUBLE PRECISION RCORX, ICORX
      DIMENSION A(1), B(1), C(1)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * X       * POINT DE DEPART DE LA METHODE DE NEWTON-RAPHSON        *
C     * NA      * DEGRE DU PREMIER POLYNOME                              *
C     * A(I)    * VECTEUR DES RACINES DU PREMIER POLYNOME                *
C     * CPA     * SA CONSTANTE MULTIPLICATIVE                            *
C     * NB      * DEGRE DU SECOND POLYNOME                               *
C     * B(I)    * VECTEUR DES RACINES DU SECOND POLYNOME                 *
C     * CPB     * SA CONSTANTE MULTIPLICATIVE                            *
C     * NC      * NOMBRE DE RACINES DU POLYNOME SOMME DEJA               *
C     *         * CALCULEES                                              *
C     * C(I)    * LES RACINES EN QUESTION                                *
C     * K       * PERMET LES IMPRESSIONS INTERMEDIAIRES                  *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * X       * RACINE CALCULEE                                        *
C     * K       * INDIQUE LA NATURE DES RESULTATS OBTENUS LORS DE        *
C     *         * LA RECHERCHE D'UNE RACINE                              *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  AUTRES VARIABLES                                                *
C     *  ****************                                                *
C     *                                                                  *
C     * N1      * NOMBRE DE CORRECTIONS DE NEWTON-RAPHSON TROP           *
C     *         * IMPORTANTES                                            *
C     * N2      * NOMBRE D'ITERATIONS AU VOISINAGE D'UNE RACINE          *
C     * N3      * NOMBRE D'ITERATIONS                                    *
C     * X       * POINT COURANT                                          *
C     * P       * VALEUR DU POLYNOME SOMME AU POINT X                    *
C     * D       * VALEUR DE SA DERIVEE AU POINT X                        *
C     * CORX    * CORRECTION DE NEWTON-RAPHSON                           *
C     * CD      * TERME CORRECTIF                                        *
C     *         *                                                        *
C     ********************************************************************


```

```
C
C
C
C      INITIALISATION.
C      ---------------
       N1 = 0
       N2 = 0
       N3 = 0
       IF ( K.EQ. 0 ) PRINT 8
 8     FORMAT (1HO,////////,11X,'POINT COURANT:',39X,'CORRECTION:',//)
C
C
C      CALCUL DE LA CORRECTION DE NEWTON-RAPHSON.
C      ------------------------------------------
 1     CALL RESULT (X,A,NA,CPA,PA)
       CALL RESULT (X,B,NB,CPB,PB)
       P = PA + PB
       IF ((DABS(DREAL(P)).LT.1.D-50).AND.(DABS(DIMAG(P)).LT.1.D-50))
      1        GO TO 401
C
       CALL DERIV (X,B,NB,CPB,PB,DB)
       CALL DERIV (X,A,NA,CPA,PA,DA)
       D = DA + DB
C
       IF ( N1.EQ. 0 ) GO TO  50
C
       IF ((DABS(DREAL(D)).LT.1.D-50).AND.(DABS(DIMAG(D)).LT.1.D-50))
      1        GO TO 99
       CORX = (-1.D0,0.D0) * P/D
       GO TO 100
C
 50    CALL SIRAC (X,C,NC,1.D0,CD)
       D = (D/P) - CD
       IF ((DABS(DREAL(D)).LT.1.D-50).AND.(DABS(DIMAG(D)).LT.1.D-50))
      1        GO TO 99
       CORX = (-1.D0,0.D0) / D
 100   RCORX = DREAL(CORX)
       ICORX = DIMAG(CORX)
       IF((DABS(ICORX).GT.1.D-20).AND.
      1       (DABS(RCORX/ICORX).LE.1.D-20)) RCORX=0.D-16
       IF ((DABS(RCORX).GT.1.D-20).AND.
      1       (DABS(ICORX/RCORX).LE.1.D-20)) ICORX=0.D-16
       CORX = DCMPLX(RCORX,ICORX)
C
C
C
C      CAS D'UNE CORRECTION DE N-R. IMPORTANTE.
C      ----------------------------------------
       IF ( CDABS(CORX).LE.1.D05) GO TO 200
```

```
 99     N2 = N2 + 1
        IF (N2.EQ.6) GO TO 101
        X = X + (0.1D0,0.1D0)
        N1 = 0
        GO TO 1
C
 101    K = 1
        RETURN
C
C       DANS LE CAS CONTRAIRE.
C       ----------------------
 200    IF (K.EQ.0) PRINT 9,X,CORX
 9      FORMAT ( 2(5X,'(',D23.16,',',D23.16,')'))
        IF ((DABS(DREAL(X)).GT.1.D-50).OR.
       1        (DABS(DIMAG(X)).GT.1.D-50)) GO TO 250
        X = X + CORX
        N3 = N3 + 1
        IF (N3-50) 1, 300, 400
C
C
C       LA PRECISION OBTENUE EST INFERIEURE A 10**-5.
C       ---------------------------------------------
 250    IF (CDABS(CORX/X).LT.1.D-5) GO TO 500
        X = X + CORX
        N3 = N3 + 1
        IF ( N3 - 50 )  1, 300, 400
 400    IF (N3.LT.150) GO TO 1
        K = 0
        RETURN
C
 300    X = (-1.D0,-1.D0)
        N1 = 0
        GO TO 1
C
C
C       LA PRECISION OBTENUE EST SUPERIEURE A 10**-5.
C       ---------------------------------------------
C          IER CAS: LA PRECISION OBTENUE  EST MAXIMALE
C          ========
 500    IF (CDABS(CORX/X).GT.5.D-16 ) GO TO 600
        X = X + CORX
        IF (K.EQ.0) PRINT 9300, X
 9300   FORMAT (1HO,///,5X,'RACINES OBTENUES AVEC UNE PRECISION ',
       1        'MAXIMALE:',//,15X,'(',D23.16,',',D23.16,')')
        K = 3
        RETURN
C
C
C          2IEME CAS: LA PRECISION OBTENUEE N'EST PAS MAXIMALE.
C          ==========
```

```
151   600  N1 = N1 + 1
152        X = X + CORX
153        IF ( N1.LT.10 ) GO TO 1
154   401  IF (K.EQ.0) PRINT 9400, X
155   9400 FORMAT (1H0,///,5X,'RACINES TROUVEES SANS PRECISION MAXIMALE:',
156       1       //,15X,'(',D23.16,',',D23.16,')')
157        K = 4
158        RETURN
159        END
```

```
      SUBROUTINE RESULT (X,A,NA,CPA,PA)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DIMENSION A(1)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * X       * POINT COURANT                                          *
C     * NA      * DEGRE DU POLYNOME                                      *
C     * A(I)    * RACINES DE CE POLYNOME                                 *
C     * CPA     * SA CONSTANTE MULTIPLICATIVE                            *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * PA      * VALEUR DU POLYNOME AU POINT X                          *
C     *         *                                                        *
C     ********************************************************************


      IF (NA.GT.0) GO TO 1
      PA = CPA
      RETURN
C
    1 PA = CPA
      DO 100 I=1,NA
      PA = PA * ( X - A(I))
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DERIV (X,A,NA,CPA,PA,DA)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DIMENSION A(1)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * X       * POINT COURANT                                          *
C     * NA      * DEGRE DU POLYNOME                                      *
C     * A(I)    * RACINES DE CE POLYNOME                                 *
C     * CPA     * SA CONSTANTE MULTIPLICATIVE                            *
C     * PA      * VALEUR DU POLYNOME AU POINT X                          *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * DA      * VALEUR DE SA DERIVEE AU POINT X                        *
C     *         *                                                        *
C     ********************************************************************


      IF ( NA - 1 ) 30,20,10

   30 DA = (0.D0,0.D0)
      RETURN

   20 DA = CPA
      RETURN

   10 DO 1 I=1,NA
      IF (CDABS(X-A(I)).LT.1.D-8) GO TO 100
    1 CONTINUE
      CALL SIRAC(X,A,NA,PA,DA)
      RETURN

  100 DA = CPA
      DO 101 J=1,NA
      IF (I.EQ.J) GO TO 101
      DA = DA*(X-A(J))
  101 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SIRAC(X,A,NA,PA,AD)
      IMPLICIT COMPLEX*16 (A,B,C,D,P,X)
      DIMENSION A(1)


C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES D'ENTREE                                              *
C     *  ******************                                              *
C     *                                                                  *
C     * X       * POINT COURANT                                          *
C     * NA      * DEGRE DU POLYNOME                                      *
C     * A(I)    * RACINES DE CE POLYNOME                                 *
C     * PA      * VALEUR DU POLYNOME AU POINT X                          *
C     *         *                                                        *
C     ********************************************************************
C     *                                                                  *
C     *  VARIABLES DE SORTIE                                             *
C     *  *******************                                             *
C     *                                                                  *
C     * AD      * VALEUR DE SA DERIVEE AU POINT X                        *
C     ********************************************************************


      IF ( NA.GT.0 ) GO TO 10
      AD = (0.D0,0.D0)
      RETURN
C
   10 AD = (0.D0,0.D0)
      DO 20 I = 1, NA
      XI = X - A(I)
      IF ((DABS(DREAL(XI)).LT.1.D-50) .AND.
     1       (DABS(DIMAG(XI)).LT.1.D-50)) GO TO 30
      AD = AD + PA/XI
   20 CONTINUE
      RETURN

   30 AD = (0.D0,0.D0)
      RETURN
      END
```