VALIDITÉ DU LOGICIEL NUMÉRIQUE

SUPPORT DE COURS

Jean-Marie CHESNEAUX

Contents

Chapter 1

L'arithmétique des ordinateurs

1.1 L'arithmétique virgule flottante IEEE-754

1.1.1 Écriture normalisée d'un réel en base b

$$\forall x \in \mathbb{R}^*, x = \varepsilon.b^e.m$$
$$\text{avec } \varepsilon \in \{-1, +1\}, e \in \mathbb{Z}, m \in \left[\frac{1}{b}, 1\right[$$
$$m = \sum_{i=1}^{+\infty} a_i.b^{-i} = 0, a_1 a_2 ... a_i ...$$
$$\text{avec } a_i \in \{0, 1, ..., b-1\}$$

ε s'appelle le signe, m la mantisse et e l'exposant.

Passage de la base 10 à la base b :
$x > 0, x = E[x] + Fract[x]$

$$E[x] = b.n_0 + r_0 \quad \text{avec} \quad 0 \leq r_0 < b$$
$$n_0 = b.n_1 + r_1 \quad \text{avec} \quad 0 \leq r_1 < b$$

ceci jusqu'à ce que $n_p = 0$. Alors

$$E[x] = \sum_{i=0}^{p} r_i.b^i = r_p r_{p-1} ... r_0 \quad \text{en base b.}$$

Pour obtenir la partie fractionnaire en base b, on multiplie $Fract[x]$ par b. Soit s_1 la partie entière de ce produit, on recommence avec la partie fractionnaire du produit pour obtenir s_2 et ainsi de suite, alors :

$$Fract[x] = \sum_{i=1}^{+\infty} s_i.b^{-i} = 0, s_1 s_2 ... s_i ... \quad \text{en base b.}$$

Finalement

$$x = r_p r_{p-1} ... r_0, s_1 s_2 ... s_i ... \quad \text{en base b.}$$

Exemple : Écrire 63,735 en base 16, puis en base 2

En base 16 :
$63 = 3 \times 16 + 15$
d'où $63_{(10)} = 3F_{(16)}$

$0,735 \times 16 = 11,76$ d'où $s_1 = 11$
$0,76 \times 16 = 12,16$ d'où $s_2 = 12$
$0,16 \times 16 = 2,56$ d'où $s_3 = 2$
$0,56 \times 16 = 8,96$ d'où $s_4 = 8$ etc.
Finalement

$$63,735_{(10)} = 3F,BC28..._{(16)}$$

En base 2 :
$63 = 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0$
d'où $63_{(10)} = 111111_{(2)}$
$0,735 \times 2 = 1,47$ d'où $s_1 = 1$
$0,47 \times 2 = 0,94$ d'où $s_2 = 0$
$0,94 \times 2 = 1,88$ d'où $s_3 = 1$
$0,88 \times 2 = 1,76$ d'où $s_4 = 1$ etc.
Finalement

$$63,735_{(10)} = 111111,10111100001..._{(2)}$$

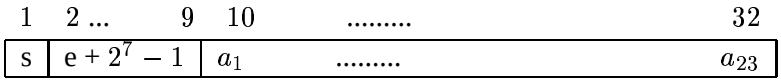
1.1.2 La norme IEEE-754

Dans l'arithmétique virgule flottante sur ordinateur, coder un réel c'est coder le triplet $\{\varepsilon, e, m\}$. Depuis 1985, une norme internationale est reconnue pour l'arithmétique virgule flottante et respectée par la quasi-totalité des constructeurs : la norme IEEE 754 [4]. Cette norme prévoit entre autres les codages **simple précision** et **double précision**. Elle utilise la base 2. Pour e et m, on utilise donc le développement binaire de ces nombres :

$$e = \sum_{i=0}^{p} b_i.2^i \text{ et } m = \sum_{i=0}^{\infty} a_i.2^{-i-1} \text{ avec } (a_i, b_i) \in \{0,1\}$$

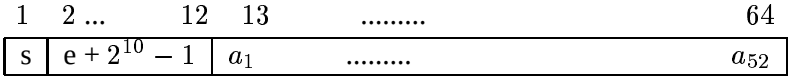
Le codage de ε tient sur un bit et vaut 0 si $x > 0$ et 1 si $x < 0$.

En simple précision, le codage de l'exposant tient sur 8 bits et est biaisé de $2^7 - 1$, la mantisse est codée sur 23 bit. On ne code pas a_0 qui vaut toujours 1 (on parle de bit caché), soit :



Codage IEEE 754 Simple précision

et pour la double précision :



Codage IEEE 754 Double précision

On appelle ensemble des flottants, noté $\mathbb{F}$, l'ensemble des réels codables sur ordinateur. Cet ensemble dépend de la précision utilisée mais c'est toujours un ensemble fini donc borné et discret. Pour être codable exactement en machine un réel doit donc

- avoir un exposant compris entre -127 et $+128$ en simple précision et entre -1023 et $+1024$ pour la double précision,

- avoir une mantisse dont le développement en base 2 ne contienne que des zéros à partir du bit 24 en simple précision et à partir du bit 53 en double précision.

Le fait que l'ensemble des nombres flottants soit borné entraîne les phénomènes de souscapacité et de surcapacité. Le fait qu'il soit discret et que les opérateurs arithmétiques ne soient pas des lois internes dans $\mathbb{F}$ entraîne le phénomène d'erreur d'arrondi.

Ainsi tout réel qui n'est pas un nombre flottant est approché par un élément de $\mathbb{F}$. Plus précisément, soit X_{min} (resp. X_{max}) le plus petit (resp. le plus grand) nombre flottant, pour tout réel x dans l'intervalle $]X_{min}, X_{max}[$, il existe $\{X^-, X^+\}$ dans $\mathbb{F}^2$ tels que $X^- < x < X^+$ et $]X^-, X^+[\cap \mathbb{F} = \Omega$

La notion de mode d'arrondi

Déterminer la règle qui, en fonction de x, fournit X^- ou X^+, c'est choisir le **mode d'arrondi** de la machine. Si $x = \varepsilon.b^e.m$ avec $m = \sum_{i=0}^{\infty} a_i.2^{-i-1}$, on a

$$X = \boxed{s} \boxed{E + 2^7 - 1} \boxed{a'_1 \quad \ldots\ldots \quad a'_{23}}$$

Déterminer le mode d'arrondi, c'est aussi concrètement déterminer le passage des a_i aux a'_i. Dans la norme IEEE 754, x est toujours représenté soit par X^- soit par X^+. On parle d'arrondi **exact** !!!

La norme IEEE 754 prévoit 4 types d'arrondi :

- L'arrondi vers zéro, x est représenté par le flottant le plus près de x compris entre x et 0. On a donc
$$a_i = a'_i \text{ et } e = E.$$

- L'arrondi au plus près, x est représenté par le flottant le plus près de x pour la distance usuelle sur $\mathbb{R}$. Ceci s'obtient en faisant
$$a'_{23} = a_{23} + a_{24}$$
avec une propagation éventuelle de la retenue. On peut donc avoir
$$E = e + \delta \text{ avec } \delta = 0 \text{ ou } 1.$$

- L'arrondi vers plus l'infini, x est systématiquement représenté par X^+. Le signe de x intervient et la relation devient
$$a'_{23} = a_{23} + \overline{s}$$
plus les mêmes comportements que précédemment ($\overline{s}$ représente la négation logique de s).

- L'arrondi vers moins l'infini, x est systématiquement représenté par X^-. On a
$$a'_{23} = a_{23} + s.$$

Cet arrondi s'effectue à chaque entrée de donnée et après chaque opération arithmétique élémentaire d'où le phénomène de propagation des erreurs d'arrondi.

1.2 Les chiffres significatifs exacts

Pour quantifier correctement la précision d'un résultat informatique, nous allons formaliser la notion de *chiffres significatifs exacts.* Parce que la base naturelle utilisée pour l'affichage est la base 10, c'est le nombre de chiffres décimaux en commun entre un résultat informatique et le résultat mathématique exact correspondant qui est important. Bien sûr, si à l'affichage ce nombre est un entier, sa définition mathématique doit être une fonction continue de R et r qui sont respectivement les résultats informatique et mathématique. On le définit par

$$C_{R,r} = \log_{10} \left| \frac{R+r}{2.(R-r)} \right| . \tag{1.1}$$

Cette définition correspond tout à fait à l'idée intuitive du nombre de chiffres décimaux significatifs en commun entre deux réels. En effet, (1.1) est équivalent à

$$|R-r| = \left| \frac{R+r}{2} \right| .10^{-C_{R,r}} . \tag{1.2}$$

Ainsi, si $C_{R,r} = 3$, l'erreur relative entre R et r est de l'ordre de 10^{-3}. R et r auront donc trois chiffres décimaux en commun.

Il ne faut toutefois pas se laisser abuser par l'écriture comme le mettent en évidence les exemples ci-dessous.

Exemple 1 $a = 1,287543$ *et* $b = 1,229835$*,*

$$\Longrightarrow \quad C_{a,b} \simeq 1,3387.$$

La première remarque est que, contrairement à l'effet visuel, il y a moins de 2 *chiffres décimaux en commun entre* a *et* b*. La différence des troisièmes chiffres décimaux significatifs affecte la validité du deuxième. Ceci peut s'illustrer encore plus nettement en arrondissant* a *et* b *au deuxième chiffre. On obtient* $a \approx 1,3$ *et* $b \approx 1,2$*.*

Exemple 2 $a = 2,45999764$ *et* $b = 2,46000123$*,*

$$\Longrightarrow \quad C_{a,b} \simeq 5,8358.$$

Bien sûr, la différence entre les 9 *et les* 0 *est illusoire. Les chiffres décimaux significatifs ne deviennent véritablement différents qu'à partir du* 6*-ième.*

Exemple 3 $a = 9,89648739$ *et* $b = 9,89650165$*,*

$$\Longrightarrow \quad C_{a,b} \simeq 5,8414.$$

Cet exemple ne contredit pas les conclusions de l'exemple 1 car le premier 9 *compte presque pour "deux" chiffres décimaux (à comparer avec* 10*). De plus, la différence réelle se situe au* 6*-ième chiffre entre le* 0 *et le* 8 *et non pas au* 5*-ième entre le* 5 *et le* 4*.*

1.3 Formalisation des erreurs d'arrondi

1.3.1 Erreur d'arrondi due aux opérateurs informatiques

- erreur due à l'opérateur d'affectation

Nous avons vu que tout réel $x \in \mathbb{R}^*$ peut s'écrire :

$$x = \varepsilon.m.b^e \text{ avec } \frac{1}{b} \leq m < 1 \tag{1.3}$$

où ε est le signe de x, m la mantisse, b la base et e l'exposant.

Si X est la représentation de x, sur un ordinateur qui utilise la base 2, on peut écrire

$$X = \varepsilon.M.2^E \text{ avec } X = x - \varepsilon.2^{E-p}.\alpha \tag{1.4}$$

$2^{E-p}.\alpha$ représente l'erreur absolue faite sur la mantisse finie M, p étant le nombre de bits de cette mantisse, le bit caché compris. α représente l'erreur d'arrondi normalisée, i.e. :

- en arrondi au plus près, $\alpha \in [-0.5, 0.5[$;
- en arrondi vers zéro, $\alpha \in [0, 1[$;
- en arrondi vers plus ou moins l'infini, $\alpha \in [-1, +1[$;

- erreur due à l'addition informatique

  Soient $\oplus$ l'addition informatique et x_1, x_2 deux réels d'images informatiques X_1 et X_2, on a

$$X_i = x_i - \varepsilon_i.2^{E_i-p}.\alpha_i \quad \text{pour} \quad i = 1, 2. \tag{1.5}$$

  et

$$X_1 \oplus X_2 = x_1 + x_2 - \varepsilon_1.2^{E_1-p}.\alpha_1 - \varepsilon_2.2^{E_2-p}.\alpha_2 - \varepsilon_3.2^{E_3-p}.\alpha_3 \tag{1.6}$$

  E_3, ε_3 et α_3 étant l'exposant, le signe et l'arrondi du résultat informatique $X_1 \oplus X_2$.

- erreur due à la soustraction

  De façon similaire, en notant $\ominus$ la soustraction informatique, on obtient

$$X_1 \ominus X_2 = x_1 - x_2 - \varepsilon_1.2^{E_1-p}.\alpha_1 + \varepsilon_2.2^{E_2-p}.\alpha_2 - \varepsilon_3.2^{E_3-p}.\alpha_3 \tag{1.7}$$

- erreur due à la multiplication informatique

  De même, soit $\otimes$ la multiplication informatique,

$$\begin{aligned} X_1 \otimes X_2 \quad = \quad & x_1.x_2 - \varepsilon_1.2^{E_1-p}.\alpha_1.x_2 - \varepsilon_2.2^{E_2-p}.\alpha_2.x_1 \\ & + \varepsilon_1.\varepsilon_2.2^{E_1+E_2-2p}.\alpha_1.\alpha_2 - \varepsilon_3.2^{E_3-p}.\alpha_3. \end{aligned} \tag{1.8}$$

  Le quatrième terme est du deuxième ordre en 2^{-p}. En négligeant ce terme du deuxième ordre, on obtient

$$X_1 \otimes X_2 = x_1.x_2 - \varepsilon_1.2^{E_1-p}.\alpha_1.x_2 - \varepsilon_2.2^{E_2-p}.\alpha_2.x_1 - \varepsilon_3.2^{E_3-p}.\alpha_3. \tag{1.9}$$

- erreur due à la division informatique

  Soit $\oslash$ la division informatique, toujours en négligeant les termes d'ordre supérieur ou égal à 2^{-2p}, on a

$$X_1 \oslash X_2 = \frac{x_1}{x_2} - \varepsilon_1.2^{E_1-p}.\frac{\alpha_1}{x_2} + \varepsilon_2.2^{E_2-p}.\alpha_2.\frac{x_1}{x_2^2} - \varepsilon_3.2^{E_3-p}.\alpha_3. \tag{1.10}$$

Remarques importantes :

Dans le cas de l'addition de termes de même signe, l'exposant E_3 du résultat est égal à $Sup\,(E_1, E_2)+\delta$, δ valant 0 ou 1. Les deux termes résultant des erreurs d'arrondi sur X_1 et X_2 sont d'un ordre de grandeur inférieur ou égal à 2^{E_3-p}. L'erreur relative sur $X_1 \oplus X_2$ reste de l'ordre de 2^{-p}. Cette opération est donc "relativement stable" en ce sens où elle ne génère pas de perte brutale de précision.

Pour la multiplication, puisque

$$E_3 = E_1 + E_2 + \delta,\ \delta \text{ valant } 0 \text{ ou } -1,$$

et que pour la division,

$$E_3 = E_1 - E_2 + \delta,\ \delta \text{ valant } 0 \text{ ou } +1,$$

les conclusions sont les mêmes.

Par contre, dans le cas de la soustraction de termes de même signe, on a seulement $E_3 = Sup\,(E_1, E_2) - k$. Si X_1 est très voisin de X_2, k peut valoir plusieurs unités alors que l'erreur absolue reste de l'ordre de $2^{Sup(E_1,E_2)-p}$ du fait de α_1 et α_2. L'erreur relative sur $X_1 \ominus X_2$ ($2^{Sup(E_1,E_2)-p-E_3}$) se trouve multipliée par 2^k. En une opération, on a perdu k bits significatifs exacts. Nous retrouvons là l'instabilité de la soustraction (cancellation) bien connue des informaticiens.

1.3.2 Erreurs d'arrondi dans un programme de calcul

Un programme de calcul est une suite ordonnée d'opérations arithmétiques. Supposons, pour simplifier, qu'il fournisse un résultat unique au bout de n opérations. Soit $r \in \mathbb{R}$ le résultat exact, l'ordinateur fournit un résultat $R \in \mathbb{F}$ entaché d'erreur. Il a été montré [2] que R peut se modéliser au premier ordre en 2^{-p} par

$$R \approx r + \sum_{i=1}^{n} g_i(d).2^{-p}.\alpha_i \tag{1.11}$$

$g_i(d)$ étant des coefficients ne dépendant que des données et α_i étant les quantités perdues lors des arrondis. De plus, on a supposé que les exposants et signes des résultats intermédiaires étaient indépendants des α_i.

Il est alors aisé de formaliser le nombre de bits significatifs exacts C_R du résultat informatique R. En effet

$$C_R \approx -\log_2 \left| \frac{R-r}{r} \right| = p - \log_2 \left| \sum_{i=1}^{n} g_i(d).\frac{\alpha_i}{r} \right| \tag{1.12}$$

Dans 1.12, la dernière quantité représente la perte de précision dans le calcul de R. Or cette quantité est indépendante de p. Nous venons d'établir le résultat suivant très important dans la pratique et trop peu connu des utilisateurs.

Théorème 1 *La perte de précision lors d'un calcul sur ordinateur est indépendante de la précision utilisée pour ce calcul.*

Il faut cependant souligner que ce résultat est lié à la validité de l'approximation au premier ordre du modèle.

Chapter 2

Les problèmes liés à l'ordinateur

2.1 Introduction

L'analyse numérique a pour but de développer des techniques de calcul permettant de déterminer des solutions numériques à des problèmes précis (résolution de systèmes linéaires, d'équations différentielles, calculs d'intégrales, etc.).

Ces techniques, appelées algorithmes, sont déterminées avec l'univers précis et rigoureux des mathématiques. Leur mise en oeuvre sur ordinateur soulève de nouveaux problèmes du fait que l'on quitte cet environnement (perte du continu, de la notion de limite etc.).

La perte du "continu" (nous avons vu au chapitre I que seul un sous-ensemble fini de réels était codable sur ordinateur), a pour conséquence la perte des opérations mathématiques exactes (addition, multiplication etc.) remplacées par des opérations approchées entraînant quasi systématiquement une erreur à chaque étape du calcul. Cela pose le problème, nouveau par rapport aux mathématiques, de la précision des résultats.

Le fait que ce sous-ensemble soit fini rend caduque toute notion de limite, d'asymptote, etc. Certains algorithmes ne sont donc même pas transposables sur ordinateur. C'est le cas pour le calcul de la somme d'une série, plus généralement pour tout résultat défini par une récurrence transfinie.

Les difficultés que soulève ce passage des mathématiques à l'informatique sont principalement de trois ordres : la capacité de mémoire, la puissance de calcul, la précision des calculs.

Si les deux premières difficultés ont été largement diminuées par les progrès de l'informatique, la dernière demeure fondamentale.

2.2 La capacité mémoire

En moyenne, la capacité mémoire double tous les 2 ans.

Actuellement, les capacités pour la mémoire vive (celle qui est la plus importante car elle limite la taille des programmes) est de l'ordre de plusieurs Megaoctets (10^6) pour la micro informatique et peut atteindre plusieurs Gigaoctets (10^9) pour les gros systèmes (DecAlpha). Malgré tout, il est toujours possible de saturer une mémoire vive : une matrice de dimension 1.000.000 (utilisé au CNES) déclarée en double précision nécessite $8\ 10^{12}$ octets pour son stockage !!!

Quant à la mémoire morte (disques durs etc.), elle se compte en milliers de Gigaoctets. Hormis quelques secteurs industriels très spécifiques (Météo, pneumatique, imagerie), les utilisateurs la considère comme *infinie*.

Aujourd'hui, la saturation d'un ordinateur vient plus d'un problème de puissance de calcul que d'un problème de capacité mémoire.

Remarque : La puissance des ordinateurs n'est pas tout. La sonde Voyager qui date du début des années 70 et qui est la seule fabrication de l'homme à avoir quitté le système solaire, a été conçue sur des micro-ordinateurs (Apple E) ayant 6000 octets de mémoire vive.

2.3 La puissance de calcul

La qualité d'un algorithme se juge exclusivement sur la précision des résultats qu'il fournit et sur la vitesse à laquelle il les fournit (les deux étant souvent liés), d'où l'importance de la notion de temps de calcul.

Même si les progrès de ces dernières années ont été considérables (on estime que la vitesse de calcul double tous les deux ans), la taille des problèmes évoluant dans le même sens, ce sera toujours un facteur de limite pour le calcul informatique.

Cela élimine même certaines méthodes numériques. Citons la méthode de calcul d'un déterminant par développement suivant les colonnes : pour un déterminant n x n, cette méthode nécessite de l'ordre de (n+1)! opérations (n!-1 additions et (n-1).n! multiplications). Les ordinateurs les plus puissants (CRAY) effectuent une opération en un temps supérieur à une picoseconde (10^{-12} secondes). Un déterminant 20 x 20 sera calculé en plus d'un an et demi. Allons plus loin et supposons que toutes les particules de l'univers ($\approx 10^{80}$) ait la puissance d'un CRAY et travaillent en parallèle, l'âge de l'univers ($\approx$ 20 milliards d'années) ne suffira pas pour calculer un déterminant 100 x 100. Un tel algorithme est donc inexploitable sur un ordinateur, même si mathématiquement, il est correct.

Un autre exemple est le calcul de la limite d'une série convergeant très lentement. Ainsi, considérons la série

$$\sum_{n=3}^{\infty} u_n \quad \text{avec} \quad u_n = \frac{1}{n.\log(n).\log^2(\log(n))}.$$

En encadrant cette série à l'aide d'intégrales, on obtient, si L est la limite cherchée :

$$\frac{1}{\log(\log(n+1))} \leq \left\| \sum_{k=3}^{n} u_k - L \right\| \leq \frac{1}{\log \| \log(n) \|}$$

Si l'on prend l'algorithme simple consistant à approcher L par une somme partielle de la série, obtenir L avec une précision de ε nécessite le calcul de $e^{e^{1/\varepsilon}}$ termes. Ainsi, pour avoir L à 10^{-1} près, il faut calculer $\sum_{k=3}^{n} u_k$ avec n de l'ordre de 10^{9566} !!! Un tel algorithme, mathématiquement correct, est inadapté dans ce cas.

Il est aisé de saturer un ordinateur. C'est donc un problème qu'il faut avoir à l'esprit pour ne pas se laisser entraîner dans des calculs interminables. Cela peut se produire pour les méthodes approchées qui dépendent d'un paramètre. Dans le cas de l'approximation d'une intégrale par la formule de la moyenne,

$$\int_a^b f(t)dt \approx \frac{1}{n+1}.\sum_{k=0}^{n} f\left(a + k.\frac{b-a}{n}\right)$$

si le temps de calcul de la formule est de 5 secondes pour n = 10, il sera supérieur à deux jours et demi pour $n = 10^6$. Il n'y a jamais que 3600 secondes dans une heure !

2.4 La précision des résultats

Notons que, alors que s'effectuaient des progrès considérables dans les domaines de mémoire et de puissance, du point de vue de la qualité numérique, rien n'a changé depuis quasiment 30 ans.

2.4.1 Conséquences théoriques

Perte des propriétés algébriques

Il y a perte de la quasi-totalité des propriétés algébriques des opérations élémentaires.

Exemple 4 *Perte de l'associativité. Sur ordinateur, en simple précision arrondi au plus près :*

$$\left(-10^{20} + 10^{20}\right) + 1 = 1$$

$$-10^{20} + \left(10^{20} + 1\right) = 0$$

Perte de la notion de continu

Les réels sont approchés par un sous ensemble fini donc discret.

Exemple 5 *La méthode de dichotomie arrêtée par le test*

$$\|f\left(u_{n+1}\right)\| \leq \varepsilon$$

ne pourra pas aboutir quel que soit ε.

Le théorème des valeurs intermédiaires n'est pas vérifié.

Soit $P(x) = x^3 - 111.x^2 + 1100.x - 3000$

Pour $x = 100,33469$: $P(x) = -2,345...10^{-2}$. *Pour son successeur informatique :* $P(x) = 4,6875\ 10^{-2}$, *ceci en ADA simple précision sur Sparcstation IPC.*

Le test échoue dès que $\varepsilon \leq 10^{-2}$

Perte des notions de limites et asymptotes

Toute quantité, mathématiquement définie comme limite d'une expression en fonction d'un paramètre, ne peut être calculée sur ordinateur.

Exemple 6 *On reprend le polynôme ci-dessus. Si on calcule la dérivée de P(x) au point x = 100 par la formule :*

$$P'(100) = \lim_{h \to 0} \frac{P(100+h) - P(100)}{h}$$

La vraie valeur est P'(100) = 8900 or dès que $h \leq 10^{-9}$, *sur ordinateur P(100 + h) = P(100).*

Exemple 7 *Sur ordinateur, toute image de série à termes positifs dont le terme général tend vers 0 est convergente car stationnaire !!!*

2.4.2 Conséquences pratiques

Perte de pécision

On ne fournit pas la valeur mathématique exacte mais une valeur approchée du fait de la propagation des erreurs d'arrondi.

Il faut rechercher le plus possible les formulations stables afin d'éviter les soustractions instables.

Exemple 8 *Pour le calcul du sinus hyperbolique, l'ordre de grandeur de* x *impose la formulation.*

- *Pour $x \leq 0,1$, il faut prendre*
$$sh(x) = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{(2n+1)!}.$$
- *Pour $x > 0,1$ il faut prendre*
$$sh(x) = \frac{\exp^{x} - \exp^{-x}}{2}.$$

Exemple 9 *La bonne formulation pour les solutions d'une équation du second degré $a.x^2 + b.x + c = 0$ est*
$$\begin{cases} x_1 &= \dfrac{-b - \varepsilon_b.\sqrt{\Delta}}{2a} \\ x_2 &= \dfrac{c}{a.x_1} \end{cases}$$
où ε_b est le signe de b avec, bien sûr, $\Delta = b^2 - 4.a.c$.

Exemple 10 *Pour le calcul de la variance d'un échantillon de valeurs $R_i, i = 1, ..., N$, il vaut mieux prendre*
$$s^2 = \sum_{i=1}^{N} \left(R_i - \overline{R}\right)^2$$
au lieu de
$$s^2 = \left(\sum_{i=1}^{N} R_i^2\right) - N.\overline{R}^2$$
où $\overline{R} = \dfrac{1}{N}\sum_{i=1}^{N} R_i$.

Problèmes de débranchement

Les débranchements se présentent sous la forme :

$$Si\ (A\ \geq\ B)\ alors\quad \text{séquence 1}\quad sinon\quad \text{séquence 2.}$$

Dans bien des cas, beaucoup plus fréquents qu'on ne le croit, la différence entre A et B est non significative. Cela signifie concrètement que la réponse au test varie en fonction de la propagation des erreurs d'arrondi :

$Si\ (A\ \geq\ B)\ alors$	en maths	réponse OUI
	sur ordinateur	réponse NON

On effectue une séquence de calcul totalement différente de la trajectoire mathématique.

2.5 Conséquences sur les algorithmes numériques

Dans les algorithmes directs

Définition 1 *Un algorithme est dit direct si la valeur recherchée est obtenue au bout d'un nombre fini d'opérations.*

La méthode de Gauss, le schéma de Hörner sont des méthodes directes. Ces méthodes ne présentent pas de problèmes de traduction au sens de l'implémentation de l'algorithme. Les seules difficultés sont la perte de précision et le contrôle des débranchements.

Exemple 11 *Soit à calculer l'expression*

$$P(x,y) = 9.x^4 - y^4 + 2.y^2 \quad \textit{pour } x = 10864,\ y = 18817.$$

Le résultat exact est 1.

En utilisant l'arithmétique IEEE simple précision, on trouve $7,0815898\ 10^8$ *et en double précision, on trouve* 2.

Exemple 12 *Résolution d'une équation du second degré.*

$$0.3 * x^2 \ + \ 2.1 * x \ + 3.675 \ = \ 0$$

- *Arrondi au plus près*

  d = -3.81470E-06
  Il y a deux racines complexes conjuguées.
  *z1 = -.3500000E+01 + i * 0.9765625E-03*
  *z2 = -.3500000E+01 + i * -.9765625E-03*

- *Arrondi vers zéro*

  d = 0.
  Le discriminant est nul.
  La racine double réelle est -.3500000E+01

- *Arrondi vers plus l'infini*

  d = 3.81470E-06
  Il y a deux racines réelles distinctes.
  x1 = -.3500977E+01
  x2 = -.3499024E+01

- *Arrondi vers moins l'infini*

  d = 0.
  Le discriminant est nul.
  La racine double réelle est -.3500000E+01

La méthode de Gauss est particulièrement sensible à ces deux phénomènes.

Exemple 13 *On utilise la méthode de Gauss avec recherche partielle du pivot par colonne pour résoudre le système linéaire suivant :*

$$\begin{pmatrix} 21 & 130 & 0 & 2.1 \\ 13 & 80 & 4.74e8 & 752 \\ 0 & -0.4 & 3.9816e8 & 4.2 \\ 0 & 0 & 1.7 & 9e-9 \end{pmatrix} . X \ = \ \begin{pmatrix} 153.1 \\ 849.74 \\ 7.7816 \\ 2.6e-8 \end{pmatrix}$$

Le résultat exact est : $x^t_{sol} \ = \ (1, 1, 10^{-8}, 1)$.

En simple précision IEEE, on trouve
x_ sol(1) = 0.6261987E+02 (vraie valeur : 0.1000000E+01)
x_ sol(2) = -0.8953979E+01 (vraie valeur : 0.1000000E+01)
x_ sol(3) = 0.0000000E+00 (vraie valeur : 0.9999999E-08)
x_ sol(4) = 0.9999999E+00 (vraie valeur : 0.1000000E+01)

Lors de la réduction de la troisième colonne, $a(3,3) = 4864.0$ mais la valeur mathématique est 0 ! L'algorithme prend le mauvais débrancement car $a(3,3)$ est alors retenu comme le pivot maximal d'où les résultats.
Remarque : Ce n'est pas directement la division par un nombre "petit" qui est cause d'instabilité dans la méthode de Gauss.

Dans les algorithmes itératifs

Définition 2 *Un algorithme est dit itératif si la valeur recherchée est définie comme la limite d'une suite récurrente d'ordre 1, i.e.*

$$L = \lim_n U_n \quad avec \quad U_{n+1} = \mathcal{F}(U_n) \quad et \quad U_0 \quad donné.$$

Du point de vue précision, on trouve encore les difficultés inhérentes aux algorithmes directs. De plus, se pose un problème d'implémentation de l'algorithme car la notion de limite n'existe pas sur ordinateur.

En fait, trivialement, on calcule un U_{n_0} et on espère que $U_{n_0} \approx L$. Il faut donc choisir n_0 d'où la notion de **test d'arrêt**.

Il existe deux grandes classes de tests d'arrêt :

1. Sur l'erreur absolue $\Longrightarrow \quad \|U_n - U_{n-1}\| \leq \varepsilon$
2. Sur l'erreur relative $\Longrightarrow \quad \|U_n - U_{n-1}\| \leq \varepsilon . \|U_{n-1}\|$

Remarque : Ces deux tests n'assurent en rien la relation $U_{n_0} \approx L$. Pour s'en convaincre, il suffit de prendre $u_n = \sum_{i=1}^{n} 1/i$ puis $u_n = n$.

Il est donc essentiel de s'assurer *au minimum* de la convergence mathématique de l'algorithme.

Du point de vue informatique, toute la difficulté est dans le choix de ε. Si ε est trop grand, le processus est arrêté trop tôt et la valeur de L est très approximative. Si ε est trop petit, on demande une précision illusoire. Le test d'arrêt peut ne jamais être vérifié, il y a risque de bouclage.

Plus précisément, si ε_0 est l'erreur d'arrondi commise au voisinage de L, on ne peut pas estimer L à moins de ε_0. Par contre, dans le cas où L est un point attractif (ce qui dans la pratique est toujours le cas), les erreurs ne se cumulent pas d'une itération à l'autre, et donc, on doit pouvoir estimer L à ε_0 près.

Exemple 14 *On applique la méthode de Newton à la résolution de*

$$f(x) = x^4 - 1002.x^3 + 252001.x^2 - 501000.x + 250000 = 0$$

Cela consiste à définir la suite

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad avec \quad x_0 = 1100$$

Le test d'arrêt est $|x_n - x_{n-1}| \leq \varepsilon . |x_{n-1}|$ *. Le nombre maximal d'itérations est fixé à* 1000*. Les calculs sont effectués en double précision IEEE. La limite mathématique* L *est* 500

ε	n	x_n	$\lvert x_n - L \rvert$
10^{-2}	9	504.32455	8.5249634
10^{-3}	13	500.24738	0.5315247
10^{-4}	16	500.110	0.148773
10^{-5}	1000	499.89926	0.1088562
10^{-6}	1000	499.89926	0.1088562

La valeur optimale de ε est ici 10^{-4}.

La recherche d'un test d'arrêt optimal demande donc la mesure de l'erreur d'arrondi lors du calcul de $\mathcal{F}(U_n)$.

Dans les algorithmes approchés

Définition 3 *Un algorithme est dit approché si la valeur obtenue est une approximation de la valeur recherchée, i.e.*

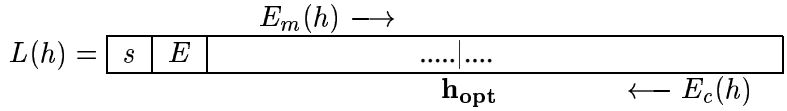
$$L = \lim_{h \to 0} L(h)$$

$L(h)$ est la valeur fournie par l'algorithme, h s'appelle le pas d'intégration.

Ces méthodes ne sont applicables que si l'on connaît *l'erreur de méthode*, i.e. une expression $E_m(h)$ telle que $|L(h) - L| \leq E_m(h)$. Bien sûr, le calcul de $L(h)$ entraîne l'apparition d'une erreur de calcul $E_c(h)$. La difficulté particulière de ce type de méthode réside dans le comportement contraire de ces deux erreurs.

$E_m(h)$ décroît quand h décroît mais $E_c(h)$ croît quand h décroît. Si h est trop grand, l'erreur de méthode est dominante mais si h est trop petit, c'est l'erreur de calcul qui devient prépondérante.

La recherche du pas optimal nécessite, là encore, la possibilité de mesurer l'erreur de calcul.



Le pas optimal correspond à $E_m(h) = E_c(h)$.

Exemple 15 *Soit la fonction :*

$$f(x) = \frac{4970.x - 4923}{4970.x^2 - 9799.x + 4830}$$

On cherche à approcher la dérivée seconde par :

$$f''(x) \approx L(h) = \frac{f(x-h) - 2.f(x) + f(x+h)}{h^2}$$

pour h "suffisamment petit".

Pour x = 1, $f''(1) = 94$. Quelques valeurs exactes sont :
$h = 10^{-4} \Rightarrow f''(1) \approx 70,78819...$
$h = 10^{-5} \Rightarrow f''(1) \approx 93,76790...$
$h = 10^{-8} \Rightarrow f''(1) \approx 94,00000...$

En double précision IEEE, on trouve :

h	$L(h)$	$\lvert L(h) - L \rvert$
10^{-1}	-9790.0020	9884.0020
10^{-2}	-917699.4887	917793.4887
10^{-3}	-2250.1982	2344.1982
10^{-4}	70.7874	23.2126
10^{-5}	93.5886	0.4114
$8\,10^{-6}$	94.0621	0.0621
10^{-6}	100.2149	6.2149
10^{-7}	5614.7087	5520.7087
10^{-8}	39790.3932	39696.3932
10^{-9}	468958.2056	468864.2056
10^{-10}	0	94

Chapter 3

La méthode CESTAC

3.1 Introduction

Il existe quatre grandes approches du problèmes des erreurs d'arrondi sur ordinateur :

- l'analyse régressive ou inverse,
- l'analyse directe,
- l'approche déterministe,
- l'approche probabiliste.

Les deux premières approches sont basées sur une analyse **a priori** des erreurs d'arrondi alors que les deux dernières utilisent directement l'arithmétique à précision finie de la machine pour faire une estimation ou majoration **a posteriori** des erreurs d'arrondi.

L'analyse régressive considère la solution informatique comme le résultat exact du même algorithme appliqué à des données perturbées. L'avantage de cette approche est de rester dans le cadre (agréable) de l'arithmétique des nombres réels. De plus cette approche permet de différencier l'erreur due au conditionnement du problème à celle due à l'instabilité de l'algorithme utilisé pour résoudre le problème en question. Cette approche, dont J.H. Wilkinson fut le précurseur [13], a donné d'excellents résultats en algèbre linéaire. Son principal inconvénient est de dépendre entièrement de l'algorithme utilisé. Il est nécessaire de faire une étude spécifique pour chaque algorithme car cette approche fait intervenir les dérivées partielles du résultat par rapport aux données et par rapport aux résultats intermédiaires. De plus, par principe, cette approche fournit un majorant de l'erreur globale et non une estimation.

L'analyse directe est moins répandue que l'analyse régressive. Cette approche consiste à décrire "pas à pas" la propagation des erreurs d'arrondi. A chaque étape, cette erreur est majorée. Dans certains problèmes, cette approche, parce qu'elle suit fidèlement l'algorithme de résolution, aboutit à des majorations plus fines que l'analyse régressive. Elle peut être également plus facile à mettre en œuvre. Elle a donné de très bons résultats et fournit parfois des informations importantes concernant le lien entre le conditionnement de certaines valeurs intermédiaires et la précision des résultats définitifs. Stummel [7] a montré, grâce à cette approche, le rapport très étroit entre la précision des pivots dans l'algorithme de Gauss et la précision de la solution.

L'approche déterministe est, comme l'approche probabiliste, une approche globale directe et conduit à une **analyse dynamique** des arrondis de calcul, i.e., elle ne dépend pas de l'algorithme et majore l'erreur en cours d'exécution du programme. A chaque étape, l'erreur est majorée en valeur absolue. Cette approche conduit à l'arithmétique d'intervalles de R. Moore surtout développée sur ordinateur par U. Kulisch et qui est la base du logiciel ACRITH, du PASCAL-XSC et du FORTRAN-XSC [8, 9, 6].

Cette suite de majorations ponctuelles conduit à la majoration finale de l'erreur. L'intérêt principal de cette approche est de fournir un cadre mathématique algébrique solide pour l'analyse des erreurs d'arrondi. Quel que soit l'intervalle résultat d'une suite de calculs effectuée en arithmétique d'intervalle, le résultat mathématique exact est toujours, par définition, dans cet intervalle. Par contre, l'étude des propriétés de l'arithmétique d'intervalle est difficile.

Le principal inconvénient de l'arithmétique d'intervalle est de ne pas pouvoir estimer l'erreur de l'arithmétique virgule flottante. Utilisée dans ce but, l'approche déterministe conduit systématiquement à une surestimation de cette erreur parfois dans des proportions importantes et ce pour deux raisons :

- elle ne tient pas compte de la compensation des erreurs d'arrondi;
- elle provoque l'effet de dispersion qui la *déconnecte* de l'arithmétique à virgule flottante.

On appelle **effet de dispersion** le phénomène suivant. Sur un ordinateur, même en utilisant l'arithmétique à virgule flottante,

$$\forall\, X \in \mathbb{F},\ \ X - X = 0 \ \ \text{et} \ \ X/X = 1.$$

Or, en arithmétique d'intervalle,

$$\text{Si}\ \ X = [1, 2],\ \ X - X = [-1, +1] \ \ \text{et} \ \ X/X = [0.5, 2].$$

Bien sûr, dans la réalité, ce phénomène se traduit de façon beaucoup plus diffuse mais reste en grande partie responsable du caractère pessimiste des estimations d'erreur fournies par l'arithmétique d'intervalle lorsque ces estimations sont comparées à l'erreur de l'arithmétique à virgule flottante.

Ceci est illustré dans les deux exemples suivants:

Exemple 16 *Calcul du déterminant de la matrice de Hilbert* $H = \left(a_{ij} = \frac{1}{i+j-1}\right)$ *de dimension 8 par la méthode de Gauss,*

Avec l'arithmétique IEEE double précision arrondie au plus près, on trouve

$$Det(H) \ = 2,7370500829639277E - 33\ .$$

Avec l'arithmétique d'intervalle, on trouve

$$Det(H) = [2,7174951443518167E - 33\ ,\ 2,7566049476830476E - 33].$$

La vraie valeur du déterminant est

$$\prod_{k=0}^{7} \frac{(k!)^3}{(n+k)!} \ \ = \ \ 2,73705011379151E - 33.$$

Le premier résultat a donc $7,95$ *chiffres décimaux significatifs exacts. L'arithmétique d'intervalle fournit* $1,84$ *chiffres décimaux significatifs.*

Exemple 17 *Calcul du produit* $\displaystyle\prod_{i=2}^{n}\left(1 - \frac{1}{i\ +\ x}\right) = \frac{1\ +\ x}{n\ +\ x}$ *avec* $x = 0,001$ *et* $n = 10^8$.

Toujours avec l'arithmétique IEEE double précision arrondie au plus près, on trouve

$$\prod_{i=2}^{n}\left(1 - \frac{1}{i\ +\ x}\right) = 1,00099999999084E - 08.$$

Avec l'arithmétique d'intervalle, on trouve

$$\prod_{i=2}^{n}\left(1-\frac{1}{i+x}\right) = [1,00099998656E-08\ ,\ 1,00100001341E-08].$$

La valeur exacte du produit est $1,00099999998999E-08$. *Le premier résultat a donc* $12,07$ *chiffres décimaux significatifs exacts alors que l'arithmétique d'intervalle fournit* $7,57$ *chiffres décimaux significatifs.*

L'arithmétique d'intervalles doit être considérée comme un nouvel environnement pour le calcul scientifique indépendant de l'arithmétique virgule flottante.

Actuellement, seule l'approche probabiliste permet de répondre à la question suivante :

Un utilisateur exécute un programme quelconque sur une machine utilisant l'arithmétique virgule flottante. Quelle est l'erreur d'arrondi engendrée lors de l'exécution de son programme ?

La méthode CESTAC (Contrôle et Estimation Stochastique des Arrondis de Calculs) a été définie par M. La Porte et J. Vignes en 1974 puis généralisée par ce dernier [10, 5].

On peut résumer très simplement l'idée majeure sur laquelle repose la méthode CESTAC. Elle consiste à exécuter plusieurs fois le même programme de calcul en propageant différemment les erreurs d'arrondi obtenant ainsi des résultats différents. La partie commune à tous les résultats représente la partie fiable, l'autre étant la partie non significative.

Il faut donc trouver une technique qui permette de générer plusieurs propagations d'arrondi pour un même calcul. Ceci est réalisé en utilisant **l'arithmétique aléatoire**.

3.2 L'arithmétique aléatoire

Tout résultat r d'une opération arithmétique qui n'est pas un flottant est encadré par deux flottants successifs R^- et R^+.

L'arithmétique aléatoire, consiste à retenir aléatoirement R^- ou R^+ avec la même probabilité $0,5$. C'est le mode d'arrondi aléatoire.

Avec ce mode d'arrondi, $\alpha_i \in [-1,+1]$. Bien sûr, ce nouveau mode d'arrondi nécessite un générateur aléatoire. Ainsi, en utilisant l'arithmétique aléatoire, un même programme exécuté N fois fournira N résultats différents.

Le but de l'arithmétique aléatoire n'est donc pas d'améliorer la précision du résultat mais seulement de faire propager différemment les erreurs d'arrondi pour, ensuite, à partir des N échantillons du résultat, pouvoir estimer son nombre de chiffres significatifs exacts.

3.3 Estimation de la précision

En exécutant N fois un programme de calcul, on obtient donc N tirages de la variable aléatoire modélisée par (1.11) où les α_i sont des variables aléatoires indépendantes équidistribuées. La distribution commune des α_i est uniforme sur $[-1,+1]$ donc centrée.

Les deux conséquences majeures sont :

1. l'espérance mathématique de la variable R est le résultat mathématique exact r,
2. la distribution de R est quasi-gaussienne.

Il s'agit donc d'estimer la moyenne d'une variable aléatoire gaussienne à partir d'un échantillon. C'est exactement le but du test de Student. Le test de Student fournit un intervalle de confiance pour l'espérance d'une gaussienne à partir d'un échantillon de cette gaussienne sous une probabilité donnée.

On sait donc que $\forall\beta \in [0,1]$, $\exists\tau_\beta \in \mathbb{R}$ tel que

$$P\left(\left|\overline{R}-r\right| \leq \frac{\tau_\beta . s}{\sqrt{N}}\right) = \beta \tag{3.1}$$

avec

$$\overline{R} = \frac{1}{N}.\sum_{i=1}^{N} R_i \quad \text{et} \quad s^2 = \frac{1}{N-1}.\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2 \tag{3.2}$$

Sous une probabilité β, le nombre de chiffres significatifs exacts de $\overline{R}$, i.e. le nombre de chiffres décimaux significatifs communs à $\overline{R}$ et à r, est majoré par

$$C_{\overline{R}} \;=\; \log_{10}\left(\frac{\sqrt{N}.\left|\overline{R}\right|}{s.\tau_\beta}\right). \tag{3.3}$$

Ainsi, l'application de la méthode CESTAC à un programme fournissant un résultat R consiste à :

1. exécuter N fois le programme en utilisant le mode d'arrondi aléatoire pour obtenir N résultats différents R_i,
2. choisir la moyenne $\overline{R}$ de cette population comme représentant informatique,
3. calculer le nombre de chiffres significatifs exacts de $\overline{R}$ en utilisant la formule (3.3).

En pratique, $N = 2$ ou 3 et $\beta = 0,95$. Pour $N = 2$, $\tau_\beta = 12,706$. Pour $N = 3$, $\tau_\beta = 4,303$.

3.4 Validation et efficacité de la méthode CESTAC

Nous présentons ici brièvement les résultats de l'étude théorique sur la validation et l'efficacité de la méthode CESTAC. Pour une discussion beaucoup plus approfondie, voir [2, 3].

3.4.1 Les hypothèses de validation

L'étude théorique a montré la validité de la méthode CESTAC sur la modélisation mathématique. Sa validité pratique repose donc entièrement sur la réalité physique ou non des hypothèses et des approximations faites dans l'étude théorique. Les deux hypothèses principales sont :

1. Les erreurs d'arrondi α_i se comportent comme des variables aléatoires indépendantes centrées équiréparties.
2. L'approximation au premier ordre en 2^{-p} dans la modélisation de R est valide.

En ce qui concerne l'hypothèse 1, remarquons tout d'abord, qu'en utilisant l'arithmétique aléatoire, les erreurs d'arrondi ne sont plus modélisées par des variables aléatoires mais sont réellement devenues des variables aléatoires. Cependant, dans la pratique, elles ne sont jamais rigoureusement centrées (i.e. de moyenne nulle). Le test de Student, parce qu'il donne toujours un intervalle de confiance de l'espérance mathématique de R, fournira un estimateur biaisé du résultat mathématique exact r.

On montre dans [1] qu'un biais de l'ordre de quelques σ - écart-type de R - entraîne une erreur inférieure à un chiffre décimal, voire à un bit, sur l'estimation du nombre de chiffres significatifs exacts. Cette robustesse s'explique principalement par la structure même de la formule $C_{\overline{R}}$ - utilisation du logarithme - et par la robustesse naturelle du test de Student.

Or dans la pratique, le biais ne dépasse jamais quelques σ. Ceci explique pourquoi, même si l'hypothèse 1 n'est pas rigoureusement satisfaite, l'estimation du nombre de chiffres significatifs exacts fourni par CESTAC n'est pas mis en défaut **si on la considère exacte à un chiffre près**.

Par contre, la légitimité de l'hypothèse 2 est fondamentale pour la validation de la méthode CESTAC. Si elle n'est pas vérifiée, R n'est plus centré sur le résultat mathématique exact r, et il apparaît un biais qui, contrairement au cas précédent, peut être prépondérant devant r.

Pour utiliser correctement la méthode CESTAC, il faut analyser ce qui peut invalider l'approximation au premier ordre. Les additions et soustractions ne génèrent pas de termes d'ordre supérieur. Par contre, les multiplications et les divisions peuvent générer des termes d'ordre supérieur [2, 3].

Soient $X_i = x_i + \varepsilon_i$ pour $i = 1, 2$. x_i sont les valeurs mathématiques exactes, X_i les valeurs informatiques et ε_i les erreurs d'arrondi faites lors du calcul des X_i. On a

$$X_1.X_2 = x_1.x_2 + x_1.\varepsilon_2 + \varepsilon_1.x_2 + \varepsilon_1.\varepsilon_2 \tag{3.4}$$

et

$$\frac{X_1}{X_2} = \frac{x_1}{x_2} + \frac{\varepsilon_1}{x_2} - \frac{x_1.\varepsilon_2}{x_2^2} + \mathcal{O}\left(\frac{\varepsilon_2}{x_2}\right) \tag{3.5}$$

Pour la multiplication, le produit $\varepsilon_1.\varepsilon_2$ devient prépondérant devant les termes du premier ordre si à la fois ε_1 et ε_2 sont prépondérants devant x_1 et x_2.

Dans le cas de la division, les termes d'ordre supérieur ou égal à 2^{-2p} du résultat resteront négligeables devant les termes du premier ordre si et seulement si ε_2 (seulement) est négligeable devant x_2. Ceci met bien en évidence le fait que l'approximation au premier ordre est de moins en moins valable au fur et à mesure que la précision de certains résultats intermédiaires décroît.

Nous venons de voir qu'elle peut être mise en défaut dans le cas de la multiplication de deux résultats non significatifs ou d'une division par un résultat non significatif.

En conclusion, la validation de la méthode CESTAC nécessitera le contrôle dynamique (i.e. en cours d'exécution) de certaines opérations arithmétiques.

3.4.2 Sur l'efficacité de la méthode CESTAC

L'efficacité pratique de la méthode repose sur deux caractéristiques :

1. le nombre de passages nécessaires au bon fonctionnement de la méthode,
2. l'influence de la probabilité de l'intervalle de confiance : 95%.

Non seulement la méthode CESTAC fonctionne très bien avec 2 ou 3 passages mais chercher à augmenter le nombre de passages est inutile. Améliorer l'estimation de 1 chiffre décimal significatif c'est diviser par 10 la longueur de l'intervalle de confiance. Cette longueur évolue en $\frac{1}{\sqrt{N}}$ où N est le nombre de tirages. Il faudrait donc multiplier par 100 le nombre de tirages pour *mathématiquement* augmenter de 1 le nombre de chiffres significatifs exacts pour une même probabilité, soit exécuter plusieurs centaines de tirages. On atteint dès lors les limites du modèle. Augmenter N dans de telles proportions revient à mettre en évidence l'erreur de modèle sans améliorer pour autant la qualité de l'estimation. Le petit nombre de tirages nécessaires à une bonne utilisation de la méthode CESTAC fait tout son intérêt pratique.

En ce qui concerne la probabilité de l'intervalle de confiance, il faut se donner une borne d'erreur dans l'approximation, par exemple de 1 chiffre significatif.

Initialement, à 95%, le résultat exact r vérifie

$$r \in \left[\overline{R} - \frac{\tau_\beta.s}{\sqrt{N}}\ ,\ \overline{R} + \frac{\tau_\beta.s}{\sqrt{N}}\right].$$

On est donc optimiste de plus de 1 chiffre décimal si

$$\left|r - \overline{R}\right| \geq \frac{10.\tau_\beta.s}{\sqrt{N}},$$

et on est pessimiste de plus de 1 chiffre décimal si

$$\left|r - \overline{R}\right| \leq \frac{\tau_\beta . s}{10.\sqrt{N}}.$$

Si l'on garde, en première approximation, l'hypothèse d'une répartition gaussienne pour R, lorsque N=3, la probabilité de fournir une estimation supérieure (cas *optimiste*) de plus de 1 chiffre à la précision réelle est de $0,00054$ et la probabilité de fournir une estimation inférieure (cas *pessimiste*) de plus de 1 chiffre à la précision réelle est de $0,29$.

En choisissant un intervalle de confiance à 95%, on préfère garantir un nombre de chiffres significatifs exacts minimal avec une très forte probabilité (0.99946) quitte à être souvent pessimiste de 1 chiffre.

3.5 Notion de zéro informatique

Dans la discussion sur la validation, nous avons vu que l'approximation au premier ordre en 2^{-p} peut brutalement être invalidée si on divise par un résultat très mal calculé ou si on multiplie entre eux deux opérandes également très mal calculées. On voit que les résultats **non significatifs** jouent un rôle particulier dans la maîtrise de la méthode CESTAC. Ceci nous amène à introduire la notion de zéro informatique au sens de J. Vignes [11] définie ci-dessous.

Définition 4 *Un résultat informatique R est un zéro informatique si $R = 0$ en étant significatif ou bien si R est quelconque mais non significatif.*

Concrètement, avec la méthode CESTAC, un résultat R, représenté par N résultats R_i, sera un zéro informatique si l'une des deux conditions suivantes est remplie.

1. $\forall i, \;\; R_i = 0$,
2. $C_{\overline{R}} \leq 0$.

Un zéro informatique est noté $\underline{0}$.

Du point de vue de la machine, un zéro informatique est un résultat que l'ordinateur, du fait de la propagation des erreurs d'arrondi de l'arithmétique à virgule flottante, ne peut distinguer de la valeur nulle. Nous verrons que dans de nombreux cas, les zéros informatiques doivent être traités sur ordinateur comme s'il s'agissait de zéros mathématiques.

3.6 Implémentation synchrone

Nous avons vu la nécessité de pouvoir détecter certains zéros informatiques lors de l'exécution d'un programme utilisant la méthode CESTAC pour assurer la validation des estimations de précision qu'elle fournit. Il faut donc pouvoir estimer le nombre de chiffres significatifs exacts de certains résultats intermédiaires. Pour cela, on doit disposer des N échantillons indispensables à l'évaluation de la formule (3.3).

Cela impose d'utiliser **l'implémentation synchrone** de la méthode CESTAC. On effectue N fois ($N = 2$ ou 3) chaque opération arithmétique en utilisant l'arithmétique aléatoire avant d'exécuter la suivante. Ainsi tout se passe comme si N programmes de calculs s'exécutaient en simultané sur N ordinateurs synchronisés. Disposant ainsi de N représentants pour tout résultat intermédiaire, on est en mesure d'estimer sa précision et donc de détecter les zéros informatiques.

De plus, cette implémentation est rendue obligatoire par l'utilisation des débranchements conditionnels inhérents à tout programme de calcul scientifique.

Si les N passages sont exécutés séquentiellement, les réponses peuvent changer d'une exécution à l'autre. Il serait dès lors absurde de chercher à estimer la précision. Comparer les résultats obtenus par les N passages n'a de sens qui si les mêmes séquences de calcul sont exécutées. Pour cette raison, il faut que la réponse au test soit la même pour tous les passages et que cette réponse tienne compte de la précision des opérandes. Nous reviendrons sur les *bonnes* définitions des relations d'ordre lorsque l'on prend en compte la précision.

Chapter 4

Introduction à l'arithmétique stochastique

4.1 Les nombres et opérations stochastiques

En se basant sur le nouveau concept de zéro informatique, on peut alors établir (et utiliser) de nouvelles définitions pour les relations d'ordre ou d'égalité sur ordinateur qui soient adaptées au calcul à précision limitée. Ceci permet de retrouver une cohérence algébrique perdue par l'arithmétique à virgule flottante.

En effet, si X et Y sont des valeurs informatiques et x et y les valeurs mathématiques exactes, au cours de l'exécution d'un programme, on sait bien que

$$X = Y \not\Rightarrow x = y \text{ et } x = y \not\Rightarrow X = Y \tag{4.1}$$

mais on oublie trop souvent qu'également

$$X \geq Y \not\Rightarrow x \geq y \text{ et } x \geq y \not\Rightarrow X \geq Y. \tag{4.2}$$

La modélisation de l'arithmétique définie par ces nouveaux concepts est appelée *arithmétique stochastique* [3]. Dans ce chapitre, il faut toujours garder à l'esprit que, derrière la modélisation mathématique, il y a la volonté de décrire qualitativement le calcul sur ordinateur avec contrôle de la précision.

Comme l'arithmétique aléatoire transforme un résultat informatique en une variable aléatoire quasi-gaussienne, on pose :

Définition 5 *L'ensemble des nombres stochastiques, noté $\mathcal{S}$, est défini comme l'ensemble des variables aléatoires gaussiennes. Un élément $X \in \mathcal{S}$ est noté*

$$X = \left(m, \sigma^2\right)$$

où m est l'espérance de X et σ son écart-type.

On peut associer à chaque nombre stochastique une précision. Si $X \in \mathcal{S}$ et $X = \left(m, \sigma^2\right)$, il existe λ_β (dépendant seulement de β) tel que

$$P\left(X \in [m - \lambda_\beta.\sigma \ , \ m + \lambda_\beta.\sigma]\right) = \beta.$$

$I_{\beta,X} = [m - \lambda_\beta.\sigma \ , \ m + \lambda_\beta.\sigma]$ est l'intervalle de confiance de m à β. Pour β = 0.95, $\lambda_\beta \approx 1.96$.

Le nombre de chiffres significatifs de X est celui commun à tous les éléments de $I_{\beta,X}$. Il est défini dans $\overline{\mathbb{R}}$ par

$$C_{\beta,X} = \log_{10}\left(\frac{\|m\|}{\lambda_\beta.\sigma}\right).$$

On associe ainsi de façon intrinsèque une précision à chaque nombre stochastique. Il faut noter que, contrairement à la définition pratique du nombre de chiffres significatifs fourni par la méthode CESTAC, le nombre de tirages N n'apparaît pas. Toutefois, pour N de l'ordre de quelques unités les deux définitions sont très voisines. Ce point est très important pour l'utilisation sur ordinateur des concepts de l'arithmétique stochastique.

Il faut maintenant définir l'ensemble des opérateurs et relations arithmétiques.

Définition 6 *Soient $X_1 = (m_1, \sigma_1^2)$ et $X_2 = (m_2, \sigma_2^2)$ éléments de $\mathcal{S}$, on définit les quatre opérations élémentaires $(s+\ ,\ s-\ ,\ s*\ ,\ s/)$ sur les nombres stochastiques par*

$$\begin{array}{lcl} X_1\ s+\ X_2 & \overset{Def}{=} & \left(m_1 + m_2\ ,\ \sigma_1^2 + \sigma_2^2\right) \\ X_1\ s-\ X_2 & \overset{Def}{=} & \left(m_1 - m_2\ ,\ \sigma_1^2 + \sigma_2^2\right) \\ X_1\ s*\ X_2 & \overset{Def}{=} & \left(m_1 * m_2\ ,\ m_2^2.\sigma_1^2 + m_1^2.\sigma_2^2\right) \\ X_1\ s/\ X_2 & \overset{Def}{=} & \left(m_1/m_2\ ,\ \left(\frac{\sigma_1}{m_2}\right)^2 + \left(\frac{m_1.\sigma_2}{m_2^2}\right)^2\right) avec\ \ m_2 \neq 0. \end{array}$$

Par définition, ces opérations sont internes dans $\mathcal{S}$. Elles correspondent aux termes du premier ordre en $\frac{\sigma}{m}$ des opérations entre deux variables aléatoires gaussiennes indépendantes.

La prochaine étape est la définition du zéro stochastique. La définition du zéro stochastique que nous adoptons est la modélisation de la notion de zéro informatique .

4.2 Les relations d'ordre stochastiques

Définition 7 *$X \in \mathcal{S}$ est un zéro stochastique, noté $\underline{0}$, si et seulement si*

$$C_{\beta,X} \leq 0\ ou\ X = (0,0).$$

Il est essentiel de toujours faire le lien entre les définitions théoriques de l'arithmétique stochastique et le calcul sur ordinateur. Concrètement, un zéro stochastique modélise un résultat informatique sur lequel l'erreur d'arrondi est au moins du même ordre de grandeur que la valeur mathématique. C'est un résultat que l'ordinateur, du fait de la précision limitée de son arithmétique, est incapable de distinguer du zéro mathématique. De plus, par définition, le zéro mathématique sera **toujours** représenté sur ordinateur par un zéro stochastique. Cette remarque triviale a des conséquences importantes. Ainsi, dans tous les algorithmes finis (i.e. lorsque le résultat est obtenu en une suite finie d'opérations), si le résultat vérifie une relation du type $\mathcal{F}(\mathcal{X}) = 0$, il est certain que, sur ordinateur, $F(X)$ sera un zéro stochastique. Une fois posée la définition du zéro stochastique, on en déduit les définitions de l'égalité stochastique et des relations d'ordre stochastiques.

Définition 8 *Soient X et Y deux éléments de $\mathcal{S}$, X est stochastiquement égal à Y, noté $X\ s=\ Y$, si et seulement si*

$$X\ s-\ Y \quad = \quad \underline{0}.$$

Définition 9 *Soient $X_1 = (m_1, \sigma_1^2)$ et $X_2 = (m_2, \sigma_2^2)$ éléments de $\mathcal{S}$, X_1 est stochastiquement strictement supérieur à X_2, noté $X_1\ s>\ X_2$, si et seulement si*

$$m_1 - m_2 > \lambda_\beta.\sqrt{\sigma_1^2 + \sigma_2^2}.$$

Définition 10 *Soient* $X_1 = (m_1, \sigma_1^2)$ *et* $X_2 = (m_2, \sigma_2^2)$ *éléments de* $\mathcal{S}$*,* X_1 *est stochastiquement supérieur ou égal à* X_2*, noté* $X_1 \; s\geq \; X_2$*, si et seulement si*

$$m_1 \geq m_2 \quad ou \quad X_1 \; s= \; X_2$$

Concrètement, deux résultats informatiques sont stochastiquement égaux si leur différence est non significative ce qui revient à dire qu'à ce moment du calcul, avec son arithmétique et sa précision, du fait des erreurs d'arrondi, l'ordinateur est incapable de les différencier. De même, l'expression *stochastiquement strictement supérieur à* correspond sur ordinateur à *significativement plus grand que.*

Ces relations vérifient les propriétés ci-dessous.

Proposition 1 *Soient X et Y deux éléments de* $\mathcal{S}$ *d'espérance x et y,*

$$x = y \quad \Rightarrow \quad X \; s= \; Y.$$

Proposition 2 $s=$ *est une relation réflexive et symétrique mais non transitive.*

Proposition 3 $X_1 \; s> \; X_2 \quad \Rightarrow \quad m_1 > m_2.$

Proposition 4 $m_1 \geq m_2 \quad \Rightarrow \quad X_1 \; s\geq \; X_2.$

Proposition 5 *La relation* $s>$ *est la négation de* $s\leq$*.*

Proposition 6 *La relation* $s>$ *est transitive.*

Proposition 7 *La relation* $s\geq$ *est réflexive, anti-symétrique mais non transitive.*

La proposition 1 est très importante car elle permet d'atteindre l'un des buts de l'arithmétique stochastique : retrouver un lien entre les opérateurs de relation sur ordinateur et les opérateurs de relation mathématiques. On retrouve ainsi beaucoup de relations algébriques mathématiques perdues par l'arithmétique virgule flottante. Par exemple, en arithmétique stochastique, on a toujours

$$(A \; s- \; B)^2 \; s= \; A^2 \; s- \; 2As* \; B \; s+ \; B^2,$$

puisque ces deux expressions sont mathématiquement égales. Il en va de même de toutes les identités remarquables, de l'associativité de l'addition ou de la multiplication, etc.

Les propositions 3 et 4 sont à comparer avec 4.1 et 4.2. Avec les définitions de l'arithmétique stochastique, on récupère un sens de l'équivalence entre les relations d'ordre sur ordinateur et les relations d'ordre mathématiques.

Si X est stochastiquement strictement supérieur à Y, on est sûr que les valeurs mathématiques correspondantes sont dans le même sens.

Les propositions 5 et 6 montrent la cohérence interne entre ces opérateurs.

Les propositions (2) et (7) marquent la différence entre l'arithmétique stochastique et l'arithmétique continue. La non transitivité de l'égalité stochastique ne doit pas surprendre. Le contrôle de la propagation des erreurs d'arrondi ne supprime pas ces erreurs. L'arithmétique de l'ordinateur n'a pas les propriétés de l'arithmétique exacte. L'arithmétique stochastique qui modélise le calcul informatique avec la prise en compte des erreurs d'arrondi doit faire apparaître ces différences et ne peut pas avoir la même structure que l'arithmétique exacte. Il est au contraire remarquable que l'arithmétique stochastique ait pu mettre en évidence les oppositions théoriques de fond entre le calcul exact et le calcul approximatif de l'ordinateur.

De plus, en identifiant la notion théorique du nombre de chiffres signficatifs d'un nombre stochastique et le nombre de chiffres significatifs exacts d'un résultat informatique obtenu par la méthode CESTAC, on peut utiliser sur ordinateur tous les concepts de l'arithmétique stochastique.

Concrètement, on assimile

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}.\left|\overline{R}\right|}{s.t_\beta}\right)$$

à

$$C_{\beta,X} = \log_{10}\left(\frac{\|m\|}{\lambda_\beta.\sigma}\right).$$

C'est ce qui a été réalisé dans le logiciel CADNA : *Control of Accuracy and Debugging for Numerical Applications* [3, 12].

Bibliography

[1] J.-M. Chesneaux et J. Vignes, Sur la robustesse de la méthode CESTAC, C.R.A.S., Paris, t.307, série 1, 1988, pp.855-860.

[2] J.-M. Chesneaux, Study of the computing accuracy by using probabilitic approach, *Contribution to Computer Arithmetic and Self-Validating Numerical Methods*, ed. C. Ulrich, (J.C. Baltzer), 1990, pp. 19-30.

[3] J.-M. Chesneaux, L'Arithmétique Stochastique et le Logiciel CADNA, Habilitation à diriger des recherches, Université Pierre et Marie Curie, Novembre 1995.

[4] IEEE Standard 754-1985 for Binary Floating-Point Arithmetic, IEEE. Reprinted in SIGPLAN 22, 2, 9-25.

[5] M. Pichat et J. Vignes, *Ingénierie du contrôle de la précision des calculs sur ordinateur*, Editions Technip, 1993.

[6] R.E. Moore, *Interval Analysis*, Englewood Cliffs, N.J.: Prentice Hall, 1966.

[7] F. Stummel, Forward analysis of Gaussian elimination - Part I and II, Numer. Math., 46, 365-415 (1985).

[8] U. Kulisch, ed.*PASCAL-SC: A Pascal extension for Scientific Computation, information manual and floppy disks*, Stuttgart: B.G. Teubner, Chichester: John Wiley & Sons, 1987.

[9] U. Kulisch and W.L. Miranker, *A new approach to scientific computation*, New-York, Academic Press, 1983.

[10] J. Vignes and M. La Porte, Error analysis in computing in: *Information Processing 74*, North-Holland, 1974.

[11] J. Vignes, Zéro mathématique et zéro informatique, La vie des Sciences, C.R. Acad. Sci., Paris, 4n^{o} 1, janvier 1987, pp. 1-13.

[12] J. Vignes, A stochastic arithmetic for reliable scientific computation, Math. Comp. Simul., 35, 1993, pp. 233-261.

[13] J. H. Wilkinson, *Rounding errors in algebraic processs*, Englewoods Cliffs, N.J., Prentice-Hall, 1963,