Méthodes d'intégration numérique

1. Soient x_1 et x_2 deux points de l'intervalle $I = [-1, 1]$ et λ_1, λ_2 deux réels. On désigne par $\mathcal{C}[-1, 1]$ l'espace vectoriel des fonctions continues sur I et à valeurs réelles. On définit :

$$T : \mathcal{C}[-1, 1] \to \mathbb{R} \text{ par } T(f) = \lambda_1 f(x_1) + \lambda_2 f(x_2).$$

   (a) Quelles conditions doivent vérifier x_1, x_2, λ_1 et λ_2 pour que T soit une méthode d'intégration sur $[-1, 1]$ exacte pour
      i. Les fonctions constantes.
      ii. Les fonctions affines.
      iii. Les polynômes de degré inférieur ou égal à 2.

   (b) Parmi les méthodes exactes pour les polynômes de degré inférieur ou égal à 2, une seule vérifie $x_1 = -x_2$. Montrer que ce choix de x_1 et de x_2 (et des λ_1, λ_2 correspondants) fournit une méthode exacte pour les polynômes de degré inférieur ou égal à 3. Expliciter la formule obtenue.

2. La formule d'intégration approchée de Simpson consiste à approximer $\displaystyle\int_a^b f(x)dx$ par

$$\int_a^b f(x)dx \simeq \frac{(b-a)}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right) = S(f).$$

   On note $E(f) = \displaystyle\int_a^b f(x)dx - S(f)$ l'erreur produite.

   (a) Quel est le plus grand $k \in \mathbb{N}$ tel que l'on ait

$$E(p) = 0, \ \forall p \ \in P_k \ ?$$

   (b) Vérifier que

$$|E(f)| \leq 2(b-a)\,\mathrm{Sup}_{x\in[a,b]}|f(x)|, \ \forall f \in \mathcal{C}^0([a, b]).$$

   (c) En utilisant un développement de Taylor, montrer qu'il existe une constante $C > 0$ telle pour tout segment $[a, b]$ et pour toute fonction $f \in \mathcal{C}^4([a, b])$ on ait :

$$|E(f)| \leq C(b-a)^5 \ \mathrm{Sup}_{x\in[a,b]}|f^{(4)}(x)|.$$

   On ne demande pas de déterminer la valeur optimale de C.

3. On se propose d'étudier la formule d'intégration approchée des trapèzes, en suivant le plan de l'exercice précédent. On rappelle que cette formule consiste à approximer $\displaystyle\int_a^b f(x)dx$ par

$$\int_a^b f(x)dx \simeq \frac{(b-a)}{2}\,(f(a) + f(b)) = T(f).$$

   On note $E(f) = \displaystyle\int_a^b f(x)dx - T(f)$ l'erreur produite.

   (a) Déterminer le plus grand $k \in \mathbb{N}$ tel que l'on ait

$$E(p) = 0, \ \forall p \ \in P_k.$$

(b) En procédant comme au (c) de l'exercice 2, déterminez les entiers p et q optimaux tels que l'on ait :

$$|E(f)| \leq C(b-a)^p \ \mathrm{Sup}_{x\in[0,1]}|f^{(q)}(x)|.$$

Ici la fonction f sera supposée aussi régulière que nécessaire.

4. Utiliser la règle des trapèzes pour estimer $I = \int_0^1 xe^{-x^2}dx$. Obtenir une majoration de l'erreur et comparer avec la valeur exacte.

5. Utiliser la méthode de Simpson pour estimer la valeur de $I = \int_0^1 (1+x^2)^{3/2}dx$ avec une précision de 10^{-5}.

6. Pour calculer une valeur approchée de $I = \int_a^b f(x)dx$ on propose la méthode suivante: remplacer f par le polynôme $p_3(x)$ qui vérifie les conditions d'interpolation suivantes: $p(a) = f(a), p'(a) = f'(a), p(b) = f(b), p'(b) = f(b)$.

   (a) Expliciter $p_3(x)$.

   (b) Obtenir la formule d'intégration numérique correspondante à la méthode précédente – la méthode des trapèzes corrigée.

   (c) Obtenir la correspondante formule d'erreur et comparer cette méthode avec les règles des trapèzes et de Simpson.

   (d) Quel est la règle composite correspondante ?

   (e) Comment peut–on calculer en utilisant la règle précédente une valeur approchée de $I = \int_0^1 e^{-x^2}dx$ avec une précision de 10^{-6} ?

7. On veut calculer une approximation de la valeur $I = \int_a^b f(x)dx$ en utilisant la règle des trapèzes :

$$T(f) = \frac{b-a}{2}\left(f(a)+f(b)\right).$$

On va déduire une nouvelle expression de l'erreur correspondante à la règle des trapèzes composites quand $f'(x)$ est intégrable sur $[a,b]$.

   (a) Montrer que

$$\int_a^b f(x)dx = \left[(b-\frac{a+b}{2})f(b) - (a-\frac{a+b}{2})f(a)\right] - \int_a^b \left(x-\frac{a+b}{2}\right)f'(x)dx$$

   (b) Montrer que

$$E(f) = \int_a^b f(x)dx - T(f) = \int_a^b K(t)f'(t)dt$$

   avec $K(t) = \frac{a+b}{2} - t$.

   (c) On considère maintenant $[a,b] = [0,1]$. Soient $T_n(f)$ la formule des trapèzes composite correspondante à une subdivision de $[0,1]$ en n sous–intervalles de même amplitude et $t_j, j = 0,\cdots,n$ les extrémités de ces sous–intervalles. Montrer que l'erreur est alors donnée par

$$E_n(f) = \int_0^1 K_n(t)f'(t)dt,$$

   avec $K_n(t)$ affine par morceaux définie par

$$K_n(t) = \frac{t_{j-1}+t_j}{2} - t, \ \text{pour } t \in [t_{j-1}, t_j], \ j = 1,\cdots,n$$

(d) Appliquer le résultat précédent pour majorer l'erreur de la formule des trapèzes composites appliquée à la fonction $f(x) = x^\alpha \ln(x)$.

8. Quadrature adaptative

Pour calculer une valeur approchée de $\int_a^b f(x)dx$ on divise l'intervalle en N sous–intervalles de même amplitude et d'extrémités x_i, x_{i+1}. On pose $I_i = \int_{x_i}^{x_{i+1}} f(x)dx$. Soit

- S_i le résultat de l'application de la méthode de Simpson à l'intervalle $[x_i, x_{i+1}]$;
- $\tilde{S}_i$ l'approximation de I_i qui résulte de l'application de la méthode de Simpson à l'intervalle $[x_i, x_i + h/2]$ et $[x_i + h/2, x_{i+1}]$.

(a) Calculer $I_i - S_i$ et $I_i - \tilde{S}_i$.

(b) En supposant que $f^{(iv)}(x)$ est approximativement constante sur l'intervalle $[x_i, x_{i+1}]$ estimer $I_i - \tilde{S}_i$.

(c) Proposer un algorithme pour le calcul d'une valeur approchée de I avec une précision ϵ en utilisant la quadrature adaptative

(d) Quel est l'avantage de cette méthode par rapport à une méthode composite ?

Annexe : arithmétique de l'ordinateur

1 Représentation des réels en virgule flottante normalisée

1.1 Les nombres machine

On rappelle tout d'abord le résultat bien connu :

Proposition 1 *Soit $b \geq 2$ un entier. Alors tout réel $x \neq 0$ se représente de manière unique par*

$$x = \pm M.\ b^E$$

où M est la mantisse normalisée ($\frac{1}{b} \leq M < 1$), b la base et E est l'exposant (entier). On peut écrire

$$M = \sum_{i=1}^{+\infty} a_i b^{-i}\ , 0 \leq a_i < b,\ a_1 \neq 0,$$

ainsi $x = \pm 0.a_1a_2a_3\cdots a_n\cdots \quad b^E$

Exemple : $311 = +0.3100\cdots\ 10^3$ $(b = 10)$, $1/3 = +0.333\cdots\ 10^0$ $(b = 10)$.

Bien évidemment, un ordinateur ne réserve pour la mantisse qu'un nombre fini r de chiffres, et un intervalle $[m_1, m_2]$ pour l'exposant. Les réels représentés sont donc de la forme (en virgule flottante normalisée)

$$\bar{x} = \pm 0.a_1a_2\cdots a_r\ \ b^E, \quad 0 \leq a_i < b,\ a_1 \neq 0,\ m_1 \leq E \leq m_2,$$

et l'ensemble des nombres machine sera noté par $F(b, r, m_1, m_2)$.

Le plus petit nombre machine positif est donné par b^{-1+m_1} ("seuil d'underflow") et le plus grand par $(1 - b^{-r})b^{m_2} \sim b^{m_2}$ ("seuil d'overflow"). Pour ne pas trop compliquer la présentation, on supposera dans la suite que ces seuils sont négligeables ($m_1 = -\infty$, $m_2 = +\infty$), et on utilisera dans les exemples le système décimal ($b = 10$). Néanmoins signalons dès maintenant que la représentation sur ordinateur est généralement en binaire ($b = 2$) ou en hexadécimal ($b = 16$).

1.2 Erreurs de représentation

Disposant de r chiffres pour la mantisse, l'ordinateur représente le nombre réel non nul en virgule flottante normalisée $x = \pm 0.a_1a_2a_3\cdots a_n\cdots \quad b^E$ par le nombre machine

$$fl(x) = \pm(0.a_1a_2a_3\cdots a_r + 0.\underbrace{0\cdots 0}_{r-1}\alpha)\ \ b^E,$$

ou l'entier α est défini *par arrondi au plus près*

$$\alpha = \begin{cases} 0 & \text{si } 0.a_{r+1}a_{r+2}\cdots \in [0, 1/2[, \\ 1 & \text{si } 0.a_{r+1}a_{r+2}\cdots \in [1/2, 1[\end{cases}$$

(pour b pair il suffit d'examiner le chiffre a_{r+1}). De plus, on a $fl(0) = 0$.

Exemple : $fl(59/990) = fl(+0.595959\cdots\ 10^{-1}) = +0.60\ 10^{-1}$ pour $r = 2$, $fl(59/990) = +0.596\ 10^{-1}$ pour $r = 3$.

Le nombre $\mu := \frac{1}{2} \cdot b^{1-t}$ est dit la *précision de la machine* (on le nomme aussi "taux d'incertitude", "ϵ machine"). Sa signification provient de la

Proposition 2 *L'erreur de représentation d'un nombre réel x par le nombre machine* $\mathrm{fl}(x)$ *est majorée par*

$$|x - \mathrm{fl}(x)| \leq \mu \cdot |x|.$$

L'exemple $fl(1+\tau) = 1 \neq fl(1+\mu) = 1 + 2\mu$ pour $0 \leq \tau < \mu$ montre que l'estimation précédente ne peut pas être améliorée.

1.3 Erreurs opératoires

Etant donnés deux nombres machine x, y, l'ordinateur ne détermine pas exactement le résultat d'une opération élémentaire $x \times y$, $\times \in \{+, -, \cdot, /\}$ mais une approximation $x \otimes y$.

Voici ce qui peut arriver avec une mantisse à $r = 3$ chiffres lorsque l'ordinateur effectue l'opération `1.02 - 0.0617`

nb de chiffres de garde	0	1	$3 = r$
Dénormalisation	$0.102\ 10^1$ $0.006\ 10^1$	$0.1020\ 10^1$ $0.0062\ 10^1$	$0.102000\ 10^1$ $0.006170\ 10^1$
Résultat intermédiaire	$0.096\ 10^1$	$0.0958\ 10^1$	$0.095830\ 10^1$
Normalisation	$0.960\ 10^0$	$0.958\ 10^0$	$0.958\ 10^0$

Pour les ordinateurs modernes on peut supposer que le processeur dispose de r chiffres de garde dans l'accumulateur. Par conséquent, pour deux nombres machine x, y on a

$$x \otimes y = fl(x \times y), \quad \times \in \{+, -, \cdot, /\}$$

Notons que généralement les opérations sur machine ne sont plus associatives ou distributives : avec $r = 3$ chiffres, nous avons par exemple (à vérifier)

$$-0.100\ 10^1 \oplus (0.101\ 10^1 \oplus 0.100\ 10^{-3}) \neq (-0.100\ 10^1 \oplus 0.101\ 10^1) \oplus 0.100\ 10^{-3}.$$

2 Conditionnement et Stabilité numérique

Dans les applications, une donnée x réelle est souvent connue de façon approchée seulement, car elle provient par exemple d'une mesure ou d'une méthode numérique. Notons par $\overline{x}$ cette approximation. On définit

$$\delta(x) = \overline{x} - x \quad \text{erreur absolue}, \quad \epsilon(x) := \frac{\overline{x} - x}{x} \quad \text{erreur relative (si } x \neq 0).$$

On dit que l'approximation $\overline{x}$ de x admet s chiffres significatifs si, en utilisant une mantisse à s chiffres, les nombres $fl(x)$ et $fl(\overline{x})$ coïncident. Par conséquent, le nombre de chiffres significatifs est une information sur l'erreur relative. En analyse numérique, on adapte souvent la convention de noter seulement les chiffres significatifs de la mantisse d'un certain résultat, pour ne pas laisser croire (comme le fait l'ordinateur) que tous les r chiffres de la mantisse du résultat sont corrects.

Exemple : Pour $x = 59/990 = +0.595959\cdots\ 10^{-1}$ et $r = 2$, l'erreur absolue sur son approximation $\overline{x} = fl(x) = +0.60\ 10^{-1}$ est donnée par $\delta(x) = +0.404040\cdots\ 10^{-3}$, et l'erreur relative par $\epsilon(x) = -0.67797\cdots\ 10^{-3}$ (moins de 0,1 pour cent). Ici, $\overline{x}$ admet 2 chiffres significatifs.

2.1 Le conditionnement d'un problème

Considérons le problème de l'évaluation d'une fonction $y = f(x)$ à partir d'un argument x donné (dans les applications on dispose généralement de plusieurs données et il faudra faire une étude pour chacune de ces quantités...). Dans un premier temps on oublie les erreurs de représentation et les erreurs opératoires : peut-on obtenir un résultat $\overline{y}$ fiable (proche du résultat exact y) si au lieu de x on dispose seulement d'une donnée $\overline{x}$ entachée d'erreur (par exemple, provenant d'une mesure).

On dit que le problème de l'évaluation de $y = f(x)$ est bien conditionné si une "petite" erreur relative sur la donnée x implique une "petite" erreur relative sur le résultat y. Ici la signification du mot "petit" dépend du contexte : une amplification par un facteur proche de 1 est sûrement très satisfaisante, un facteur 1000 est quelquefois encore acceptable, mais un facteur proche de $1/\mu$ peut mener à un résultat $\overline{y} = f(\overline{x})$ "approchant" $y = f(x)$ sans aucun chiffre significatif (et donc sans aucune utilité).

Si la fonction est dérivable en x, on peut utiliser un développement limité de f : au premier ordre, le facteur d'amplification de l'erreur relative (en module) est donné par la quantité

$$\kappa(f, x) := \Big|\frac{x \cdot f'(x)}{f(x)}\Big|$$

appelée conditionnement de f en x. Notons que le conditionnement est inhèrant au problème.

Exemple : Considérons le problème de l'évaluation de $y = f(x) = 1 - \cos(x)$ en $x = 1/30$. Ici le conditionnement est donné par $\kappa(f, x) \sim 1.9998\cdots$. On peut donc conclure qu'une "petite" erreur relative sur notre donnée mène à un résultat fiable (avec une erreur relative multipliée approximativement par deux). Effectivement,

$$\overline{x} = +0.33\,10^{-1} \quad \text{donne} \quad \frac{\overline{x} - x}{x} \sim 0.01, \quad \frac{\overline{y} - y}{y} = \frac{f(\overline{x}) - f(x)}{f(x)} \sim 0.02.$$

2.2 La stabilité numérique d'un algorithme

On a vu au chapitre précédent qu'un algorithme d'évaluation d'une fonction f en x peut donner des résultats fiables seulement si le problème d'évaluation est bien conditionné. Cependant, un bon conditionnement n'est pas suffisant : sur ordinateur, il faut tenir compte des erreurs de représentation et des erreurs opératoires qui peuvent être amplifiées dans les opérations suivantes. Par conséquent, différents algorithmes peuvent donner des résultats plus ou moins valables, comme on le voit dans l'exemple suivant.

Exemple : Reprenons le problème de l'évaluation de $f(x) = 1 - \cos(x)$ en $x = 1/30$, qui admet comme réponse exacte $y = f(1/30) = 0.5555\cdots\ 10^{-3}$. On travaille avec une mantisse à $r = 3$ chiffres et on dispose de l'approximation $\overline{x} = 0.33\ 10^{-1}$ de x, avec une erreur relative d'un pour cent. Supposons également que l'ordinateur sache évaluer les fonctions trigonométriques. Les deux différents algorithmes proposés sont basés sur l'identité

$$1 - \cos(x) = \frac{\sin^2(x)}{1 + \cos(x)}.$$

Dans le premier algorithme on calcule les résultats intermédiaires

$$x_1 = fl(\cos(\overline{x})) = 0.999\ 10^{0}, \quad \overline{y} = fl(1) \ominus x_1 = 0.100\ 10^{-2},$$

avec une erreur relative $|\overline{y} - y|/|y| \sim 0.26$: on obtient une amplification par le facteur 26. On note que $\overline{y}$ n'admet aucun chiffre significatif.

Dans le second algorithme on calcule les résultats intermédiaires

$$x_1 = fl(\sin(\overline{x})) = 0.330\ 10^{-1}, \quad x_2 = x_1 \odot x_1 = 0.109\ 10^{-2}, \quad x_3 = fl(\cos(\overline{x})) = 0.999\ 10^{0},$$
$$x_4 = fl(1) \oplus x_3 = 0.200\ 10^{1}, \quad \overline{y} = x_2 \oslash x_4 = 0.545\ 10^{-3},$$

avec une erreur relative $|\overline{y} - y|/|y| \sim 0.018$ beaucoup plus petite (proche de la valeur optimale prédite par le conditionnement).

Définition 1 *On dit qu'un algorithme est numériquement stable si une "petite" erreur relative sur les données implique une "petite" erreur relative sur le résultat.*

L'étude de la stabilité numérique d'un algorithme est généralement difficile. Au cours de l'exécution d'un algorithme, l'ordinateur calcule (au moins théoriquement) des quantités auxiliaires $x_0 = x, x_1, ..., x_{k-1}, x_k = f(x)$. Si on exprime le résultat final $y = f(x)$ en fonction d'une quantité auxiliaire $y = f_j(x_j)$, alors, au premier ordre, le plus grand des conditionnements $\kappa(f_j, x_j)$ peut indiquer si notre algorithme est numériquement stable.

Souvent, on se contente de vérifier si dans une opération élémentaire on a une perte importante de chiffres significatifs (c'est-à-dire une amplification importante de l'erreur relative). Supposons qu'on connaisse des approximations $\overline{x}$ et $\overline{y}$ des nombres réels x et y, avec comme erreurs absolues $\delta(x) = \overline{x} - x$ et $\delta(y) = \overline{y} - y$. Alors pour une opération élémentaire $\times \in \{+, -, \cdot, /\}$

$$\delta(x \times y) := \overline{x} \otimes \overline{y} - x \times y = \underbrace{\overline{x} \otimes \overline{y} - \overline{x} \times \overline{y}}_{\text{erreur opératoire}} + \underbrace{\overline{x} \times \overline{y} - x \times y}_{\text{erreur transmise}}.$$

En utilisant notre estimation pour l'erreur opératoire et en négligeant tout terme quadratique (les termes d'ordre 2 ou plus en $\delta(x), \delta(y), \mu$), on obtient le résultat suivant :

Proposition 3 *Avec les notations précédentes, on a*

$$\begin{array}{ll}
|\delta(x+y)| \leq |\delta(x) + \delta(y)| + \mu|x+y| & + \textit{ terme négligeable} \\
|\delta(x-y)| \leq |\delta(x) - \delta(y)| + \mu|x-y| & + \textit{ terme négligeable} \\
|\frac{\delta(x \cdot y)}{(x \cdot y)}| \leq |\frac{\delta(x)}{x} + \frac{\delta(y)}{y}| + \mu & + \textit{ terme négligeable} \\
|\frac{\delta(x/y)}{(x/y)}| \leq |\frac{\delta(x)}{x} - \frac{\delta(y)}{y}| + \mu & + \textit{ terme négligeable}
\end{array}$$

Si on oublie momentanément l'erreur opératoire, on peut résumer la propriété précédente comme suit : l'erreur absolue est additive pour les opérations $+$ et $-$ et l'erreur relative est additive pour les opérations $\cdot$ et $/$. Cependant, pour l'addition ou la soustraction on peut observer quelquefois une perte importante de chiffres significatifs, comme on va le voir.

2.3 Quelques erreurs survenant au cours de calculs

2.3.1 Erreur de cancellation

L'erreur de cancellation intervient lorsque l'on soustrait deux nombres x et y proches. Soient les nombres machine $\overline{x}$ et $\overline{y}$ des approximations de x et y, avec s chiffres significatifs. Alors l'approximation $\overline{x} \ominus \overline{y}$ de $x - y$ ne peut avoir s chiffres significatifs que si x et y n'ont pas de chiffres en commun. Dans le cas contraire on observe une perte de chiffres significatifs et on parle d'une erreur de cancellation. Voici un exemple :

Exemple : Soient $x = 0.76545421\ 10^1$ et $y = 0.76544200\ 10^1$, et $\overline{x} = fl(x)$, $\overline{y} = fl(\overline{y})$ obtenus en travaillant avec une mantisse à $r = 7$ chiffres (c'est–à–dire avec des erreurs relatives $|\epsilon(x)| \sim 1.3\ 10^{-8} \leq \mu = 0.5\ 10^{-6}$, et $\epsilon(y) = 0$). L'ordinateur donne le résultat

$$\overline{x} \ominus \overline{y} = 0.1220000\ 10^{-3}, \quad \text{l'erreur relative est} \quad \frac{(\overline{x} \ominus \overline{y}) - (x - y)}{x - y} \sim -0.819\ 10^{-3},$$

c'est-à-dire que l'erreur relative est multipliée par un facteur 63000. Effectivement, on vérifie que $\overline{x} \ominus \overline{y}$ admet seulement 3 chiffres significatifs (les quatre derniers chiffres ont été obtenus par le processus de normalisation) et on a perdu 4 chiffres significatifs.

Une autre erreur de cancellation a été rencontrée dans le premier algorithme de l'exemple du chapitre précédent. Comme moralité on peut retenir (voir aussi le deuxième algorithme de cet exemple) :

$\Longrightarrow$ *éviter de soustraire des quantités voisines.*

2.3.2 Effet de cumul

Cet effet apparaît lors de l'addition de plusieurs nombres ayant des ordres de grandeur différents.

Exemple : Plaçons-nous dans une arithmétique à 4 chiffres. Soient $\overline{x} = x = 0.1145\ 10^1$ et $\overline{y} = y = 0.3112\ 10^{-2}$. Si on demande à l'ordinateur d'additionner ces deux nombres, on obtient

$$\overline{x} \oplus \overline{y} = 0.1148\ 10^1, \quad \text{l'erreur relative est} \quad \frac{(\overline{x} \oplus \overline{y}) - (x+y)}{x+y} \sim 10^{-4}.$$

En effet,, l'approximation $\overline{x} \oplus \overline{y}$ de $x+y$ admet le maximum de 4 chiffres significatifs. Néanmoins, si on additionne à cent reprises y au résultat obtenu dans l'itération précédente, on obtient

$$\underbrace{((\cdots(}_{100 \text{ fois}} (\overline{x} \underbrace{\oplus \overline{y}) \oplus \overline{y}) \cdots \oplus \overline{y})}_{100 \text{ fois}} = 0.1448\ 10^1 \neq \overline{x} \oplus (\underbrace{(\cdots(\overline{y} \oplus \overline{y}) \oplus \overline{y}) \cdots \oplus \overline{y})}_{100 \text{ fois}} = 0.1455\ 10^1.$$

Ici la quantité à droite admet 3 chiffres significatifs, mais la quantité à gauche un seul chiffre significatif. Effectivement, dans l'expression de gauche les "petites" erreurs opératoires s'accumulent pour finalement jouer un rôle assez important.

Pour éviter un tel phénomène, il faudra
$\Longrightarrow$ *regrouper les calculs par ordres de grandeur semblables.*

L'effet de cumul peut être particulièrement important lorsque l'on fait un cumul simple (par exemple, pour évaluer une somme) de la forme

$$S_{n+1} = S_n + a_{n+1}.$$

A partir d'un certain rang, n, a_{n+1} deviendra petit devant S_n et à partir donc d'un certain rang (procédure de dénormalisation), on aura sur ordinateur

$$S_{n+1} = S_n \oplus a_{n+1} = S_n.$$

Par un tel procédé, même la somme harmonique devient convergente sur ordinateur...

3 Quelques critères pratiques

3.1 Calcul de série

3.1.1 Critères analytiques

Il faut évidemment, avant tout, exploiter les propriétés éventuelles de la série $\sum_{i=1}^{+\infty} a_i$.

Si par exemple la série est alternée ($a_i = (-1)^i b_i,\ b_i > 0, \quad b_{i+1} < b_i, \forall i \geq 1$), on vérifie aisément que $|S - S_n| \leq b_{n+1}$. On pourra alors déterminer à l'avance, pour ϵ donné, la valeur de n telle que $|S - S_n| < \epsilon$.

3.1.2 Critères numériques

Bien évidemment, on ne dispose pas toujours de critère d'arrêt et *a fortiori* pas d'estimation d'erreur (ce qui implique que l'on somme à l'endroit en général). On peut néanmoins arrêter les itérations lorsque la quantité ajoutée est considérée comme "petite" i.e. si

$$\frac{|a_n|}{|S_n|} < \epsilon, \quad \text{avec } \epsilon \text{ (plus grand que la précision machine) un seuil donné.}$$

Si on dispose d'une estimation d'erreur (de $|S - S_n|$), il convient de sommer la série "à l'envers", pour éviter les effets de cumul, comme le montre l'exemple suivant :

Application : $S_n = 1 - \frac{1}{2} + \frac{1}{3} + \cdots + \frac{(-1)^{n+1}}{n}$. Calculer (en simple précision) cette série en partant du début, en partant de la fin (en regroupant éventuellement les termes 2 par 2 pour éviter une cancellation), avec une précision de 10^{-6} et comparer le résultat obtenu avec `log(2.0)`.

3.2 Les suites

3.2.1 Critères analytiques

Même remarque que pour le cas des séries.

3.2.2 Critères numériques

On peut considérer les critères $\frac{|x_{n+1} - x_n|}{|x_n|} < \epsilon$

3.2.3 Autres tests

Lorsque l'on met en œuvre une méthode itérative, il faut se donner un nombre maximal d'itérations au delà duquel on décide qu'il est inutile de poursuivre le processus. En effet, dans le cas contraire, une erreur de programmation peut conduire à une boucle effectuée indéfiniment.

3.3 Contrôles *a posteriori*

Prendre des précautions dans la construction du programme ne suffit pas toujours. Il faut pouvoir analyser les résultats obtenus, ce qui permet de détecter des erreurs (s'il y en a).

3.3.1 Nombre de chiffres exacts

Soit x_n une suite convergente vers x^*. On mesure le nombre de chiffres décimaux exacts par

$$e(x_n, x^*) = \log_{10}\left(\frac{|x^*|}{|x_n - x^*|}\right).$$

Le nombre de chiffres exacts gagnés de x_n à x_{n+1} est alors donné par

$$d_n = e(x_{n+1}, x^*) - e(x_n, x^*) = \log_{10}\frac{|x_n - x^*|}{|x_{n+1} - x^*|}.$$

Si x_n est une suite d'ordre r i.e. si

$$0 < \underline{lim}\frac{|x_{n+1} - x^*|}{|x_n - x^*|^r} \leq \overline{lim}\frac{|x_{n+1} - x^*|}{|x_n - x^*|^r} < +\infty$$

alors le nombre de chiffres exacts est à peu près multiplié par r à chaque itération.

3.3.2 Conclusion

Il convient d'être prudent quant à la validité de résultats numériques : la précision affichée par l'ordinateur est souvent (ou même presque toujours) illusoire.

On pourra consulter avec profit l'ouvrage suivant :

C. Brezinski, *Algorithmique numérique*, Ellipse, Paris