Analyse numérique : les erreurs numériques

Dans le cadre de l'étude des problèmes numériques lors de l'utilisation d'outils liés aux calculs numériques, tels que Matlab ou Scilab, nous allons nous intéresser aux erreurs dues à la manière dont les nombres sont codés et manipulés par l'ordinateur.

On s'attendrait en effet de la part d'un ordinateur que les résultats qu'il renvoient soient absolument exacts. Cependant, le fait est que les limitations techniques entraînées par la nécessité de stocker des grandeurs en mémoire, dans une place limitée, empêchent l'utilisateur d'obtenir une précision infinie lorsqu'il effectue des calculs.

Nous allons voir que les erreurs numériques sont de plusieurs types et ont différentes causes. Nous nous intéresserons d'abord à la précision de la machine, puis aux erreurs dite « d'annulation ». Nous traiterons enfin des problèmes de propagation d'erreur (ou *smearing*).

I. Précision de la machine

On peut d'abord s'interroger sur la plus petite différence ε pouvant exister entre deux réels.

Étant donné que dans la norme VFN (Virgule Flottante Normalisée, IEEE 754), un flottant $x=\pm y\times 2^z$ est représenté en mémoire par 64 bits dans lesquels :

- le premier représente le signe (0 pour positif, 1 pour négatif)
- les 11 suivants représentent l'exposant codé en binaire (valant donc entre -1023 et +1024)
- les 52 derniers étant la mantisse, codée en binaire, comprise entre 1 et 2

on peut en déduire que le plus petit nombre représentable strictement supérieur à 1 est codé par :

0 01111111111 0000000000000000000000000000000000000000000000000001

puisque $1=1{,}0\times 2^0$ est codé par :

0 01111111111 0000000000000000000000000000000000000000000000000000

La différence ε vaut donc : $\varepsilon=2^{-52}\approx 2{,}22044\times 10^{-16}$

Cela signifie que quelle que soit l'ordre de grandeur d'un nombre manipulé avec Scilab, celui-ci n'aura jamais plus de 16 décimales exactes.

Sous Scilab, cette valeur est stockée dans %eps :

```
-->%eps
 %eps =
    2.220E-16
```

II. Erreurs d'annulation

Lorsque l'on effectue certaines opérations, par exemple la soustraction de deux nombres petits et proches, on voit apparaître ce qui semble une aberration. Ainsi :

```
-->1.234567899E-11-1.234567888E-11
 ans  =
    1.000000000577265217E-19
```

On devrait dans l'idéal obtenir $1{,}000000000000000000.10^{-19}$

On peut expliquer ce phénomène par le fait que l'on ne calcule pas en base 10. En effet, le stockage en binaire constitue une approximation des nombres entrés sous forme décimale :

$$\tilde{a}=a(1+\Delta a);\tilde{b}=b(1+\Delta b);\tilde{x}=\tilde{a}-\tilde{b};$$

Par conséquent : $\left|\frac{\tilde{x}-x}{x}\right|=\left|\frac{a\Delta a-b\Delta b}{a-b}\right|$ et ce nombre peut être très grand si a et b sont très proches.

Si l'on veut estimer l'évolution de l'erreur d'annulation lors de l'application d'une fonction, on peut faire le calcul suivant :

$$y=f(x);\left|\frac{\Delta y}{y}\right|=\left|\frac{\tilde{y}-y}{y}\right|=\left|\frac{f(\tilde{x})-f(x)}{f(x)}\right|=\left|\frac{f(x(1+\Delta x))-f(x)}{f(x)}\right|\approx\left|\frac{f(x)+xf'(x)-f(x)}{f(x)}\right|=\left|\frac{xf'(x)}{f(x)}\right|$$

On obtient ainsi un critère de stabilité : il faut que : $\left|\frac{xf'(x)}{f(x)}\right|\ll 1$ pour que l'erreur n'influence pas trop le résultat.

Somme d'une série

On peut encore constater des erreurs importantes lors de calculs impliquant des nombres d'ordres de grandeur très différent. Par exemple, la somme de la série suivante :

$$S_n=1+\sum_{k=1}^{n}\frac{1}{k(k+1)}=2-\frac{1}{n+1}=S_{th}$$

On utilise le programme suivant :

```
n=1000;
Sn=0;
Tn=1;
for k=1:n
  Sn=Sn+1/(k*(k+1));
  Tn=Tn+1/(k*(k+1));
  k=k+1;
end
Sth=2-1/(n+1)
Sn=Sn+1
Tn
erreur1=abs((Sn-Sth)/Sth)
erreur2=abs((Tn-Sth)/Sth)
```

Qui donne le résultat :

```
Sn  =
   1.999001
Tn  =
   1.999001
erreur1  =
   3.332E-16
erreur2  =
   1.555E-15
```

Dans ce cas, pour $n=1000$, la somme effectuée avec le membre de gauche fait apparaître une erreur de $1{,}555.10^{-15}$ alors que l'ajout de 1 à la somme des termes seuls ne produit une erreur que de $3{,}332.10^{-16}$, soit 10 fois plus petite.

On explique cela par le fait que le second calcul implique des termes plus petits et plus proches les uns des autres avant d'ajouter le 1 final. Le premier en revanche ajoute des termes très petits à un total déjà de l'ordre de grandeur de 1.

III. Propagation d'erreurs

Enfin, la propagation d'erreurs (ou *smearing*) est la perte de chiffres significatifs due à des erreurs d'arrondis qui se cumulent.

On peut s'en apercevoir dans le programme suivant, qui calcule $\exp(x)$ de deux manières différentes, d'une part avec la fonction exponentielle du logiciel, et d'autre part en utilisant la formule de Taylor-intégrale :

$$\exp(x)=\sum_{k=0}^{n}\frac{x^k}{k!}+\int_0^x \exp(t)\frac{t^n}{n!}dt$$

```
n=1 ; x=-20 ; ex=1 ; tol=1e-3 ; un=x ;
while abs(un)/exp(x) >= tol & n<100000
  ex=ex+un;
  n=n+1;
  un=un*x/n;
end
ex
expx=exp(x)
erreur=abs((ex-exp(x))/exp(x))
n
```

L'exécution donne :

```
ex  =
   5.623089322732710410E-09
expx  =
   2.061153622438557870E-09
erreur  =
   1.728127230070320763E+00
n  =
   7.500000000000000000E+01
```

On peut remarquer qu'au bout de 75 itérations, l'ajout d'autres termes u_n ne contribue presque plus au calcul de $\exp(x)$ (les nouveaux termes étant inférieurs à un millième de la somme). On a donc fait un calcul de $\exp(x)$ censé être précis au millième et pourtant, l'erreur relative calculée à ce point est énorme : de l'ordre de grandeur de la valeur à calculer, puisque le rapport de l'écart sur la valeur théorique vaut environ 1,7 !

Plusieurs solutions sont envisageables pour effectuer ce même calcul sans une erreur trop grande. On peut ainsi :

- regrouper les termes positifs et négatifs (les $(-20)^i$ avec i pair ou impair)
- calculer $\exp(20)$ puis utiliser le fait que $\exp(-20)=\frac{1}{\exp(20)}$

De telles optimisations minimiseront le nombre d'opérations effectuées sur des nombres très différents et donc le nombre d'erreurs d'arrondis.

Conclusion

Nous avons vu que le simple fait de déplacer la somme d'un terme avant ou après un calcul pouvait faire varier l'erreur totale d'un facteur 10. Dans d'autres cas, l'erreur relative atteint une telle taille que le résultat du calcul est simplement faux dès son premier chiffre.

On peut donc affirmer que si l'on souhaite obtenir les calculs les plus précis possibles avec l'outil informatique, il convient de regarder d'assez près les calculs que l'on effectue et de comprendre comment l'ordinateur va les effectuer.

On pourra ainsi repérer les expressions dans lesquelles des erreurs d'arrondis ou de trop grandes différences d'ordre de grandeur pourront apparaître et ainsi optimiser l'ordre des calculs afin d'augmenter la précision de ceux-ci.