Compléments de programmation
Précision et erreurs

D. Schaub

7 février 2012

Commençons par un exemple classique de récursivité

La fonction factorielle

```
function x=fact(n)
  if n<=1 then
    x=1
  else
    x=n*fact(n-1)
  end
endfunction
```

La programmation récursive est rarement efficace d'un point de vue algorithmique : c'est lent et très consommateur de mémoire. Par contre, elle permet d'écrire des programmes très courts et simples.

Chronomètre

Exemple : `timer() ; inv(rand(100,100)) ; timer()`

Input

Regardons le fichier script.sce suivant

```
A=input("Entrer un vecteur ou une matrice")
n=evstr(x_dialog("Entrer un nombre"))
sum(A.^n)
```

La commande input
interrompt le script en affichant le texte indiqué et attend que l'utilisateur rentre une expression suivie de la touche entrée.
Le programme reprend alors en utilisant la valeur entrée.

La commande *x_dialog* est identique sauf son interface plus graphique.

Break

Cette instruction interrompt le programme

Peu intéressant

```
function E=f(x)
  E=0
  while %t
    if E+1>x
      break
    end
    E=E+1
  end
endfunction
```

Que donne $f(x)$? A quoi sert la commande while %t ?

Error

Destinée à interrompre un programme en affichant un message d'erreur

Calcul de racine

```
function y=racine(x)
  if x >= 0 then
    y=sqrt(x)
  else
    error('le nombre est negatif')
  end
endfunction
```

Les nombres réels ne peuvent être représentés de manière **exacte**

En *Scilab*, un nombre est toujours approché

Problème : Gérer les erreurs en majorant l'erreur commise !

Lorsqu'on représente un nombre sous la forme

$$x \cong \pm m \times 10^p$$

avec une mantisse $1 \leq m = a_0, a_1 \cdots a_N < 10$,
l'incertitude sur x porte sur un terme en $10^{-(N+1)}$, terme majoré par 10, par conséquent, l'erreur relative commise est

$$\frac{\Delta x}{x} = \frac{\Delta m}{m} \leq \frac{10}{10^{N+1}} = 10^{-N}$$

La mantisse est un nombre fixe de $N+1$ termes (par exemple 16 dans notre cas) significatifs. Ce nombre correspond à un certain nombre de bits réservés (si le nombre est écrit en base 2, cela correspond précisément à $N+1$ bits).
Il n'y a pas **unicité** de l'écriture en virgule flottante

$$3.1416 = 31416 \times 10^{-4} = 0.31416 \times 10$$

sauf si l'on veut "normaliser" en imposant 0 comme seul chiffre avant la virgule ou en mettant un seul chiffre avant la virgule.

La mantisse est un nombre fixe de $N+1$ termes (par exemple 16 dans notre cas) significatifs. Ce nombre correspond à un certain nombre de bits réservés (si le nombre est écrit en base 2, cela correspond précisément à $N+1$ bits).
Il n'y a pas **unicité** de l'écriture en virgule flottante

$$3.1416 = 31416 \times 10^{-4} = 0.31416 \times 10$$

sauf si l'on veut "normaliser" en imposant 0 comme seul chiffre avant la virgule ou en mettant un seul chiffre avant la virgule.

De nombreux problèmes sont liés à cette représentation. Ainsi, par exemple, la non-associativité. L'addition vérifie normalement

$$(x+y)+z = x+(y+z) \ldots \text{ or}$$

La mantisse est un nombre fixe de $N+1$ termes (par exemple 16 dans notre cas) significatifs. Ce nombre correspond à un certain nombre de bits réservés (si le nombre est écrit en base 2, cela correspond précisément à $N+1$ bits).
Il n'y a pas **unicité** de l'écriture en virgule flottante

$$3.1416 = 31416 \times 10^{-4} = 0.31416 \times 10$$

sauf si l'on veut "normaliser" en imposant 0 comme seul chiffre avant la virgule ou en mettant un seul chiffre avant la virgule.

De nombreux problèmes sont liés à cette représentation. Ainsi, par exemple, la non-associativité. L'addition vérifie normalement $(x+y)+z = x+(y+z)$... or

Problèmes d'arrondis

Posons $x = 8.22$, $y = 0.00317$, $z = 0.00482$ et arrondissons les résultats à 2 décimales.
On a : $x+y \cong 8.22$, d'où $(x+y)+z \cong 8.22+0.00 = 8.22$
Mais : $y+z \cong 0.01$, d'où $x+(y+z) \cong 8.22+0.01 = 8.23$

Il existe des règles de majoration d'erreurs, ainsi

$$\Delta(x+y) \leq \Delta x + \Delta y + 10^{-16}(|x|+|y|)$$
$$\Delta(xy) \leq |x|\Delta y + |y|\Delta x + 10^{-16}|xy|$$

Remarque : *Si x est connu avec une précision Δx, alors $f(x)$ est connu avec une précision $|f'(x)|\Delta x$.*
Mais, dans notre pratique, nous nous contenterons en général de supposer que chaque opération induit une erreur de 10^{-16}. Ainsi une succession de 5000 opérations (*somme des 5000 premiers termes d'une série*, par exemple) donne une erreur maximale de $5000 \times 10^{-16} = 5 \times 10^{-13}$.

Remarque : il peut aussi y avoir des phénomènes de compensation des erreurs (exemple : calcul de e^{-10} par $e^{-10} = \sum_{k=0}^{n}(-1)^k \frac{10^k}{k!}$)

Dans des calculs récurrents ou itératifs, l'erreur $E(k+1)$ au cran $k+1$ peut-être de la forme $10 \times E(k)$. Si $E(1) = 10^{-16}$, $E(2) = 10 \times 10^{-16}, \ldots, E(17) = 10^{16} \times 10^{-16} = 1 ! ! !$

Dès lors, lorsque k devient grand, le résultat n'a plus aucune signification !

Il faut être très vigilant dès lors que l'on fait du calcul approché, surtout dans la réalité physique : calculs de contraintes, de vitesse, de poids, etc...
Une fusée ARIANE 5 (en 1996) a été victime du problème de représentabilité des nombres en virgule flottante ! ! !