Tout petit Cours de Mathématiques Appliquées à l'usage des Physiciens utilisateurs de l'Informatique

Gérard A. Langlet

COURS Numéro 1. De l'Erreur

(*Errare humanum est, perseverare diabolicum*)

Calculons les puissances de 2 à partir de 50 sur l'ordinateur le plus courant, un compatible PC, qui permet l'affichage des nombres avec 17 chiffres décimaux au maximum[1] :

```
    2*50 51 52 53
1125899906842624 2251799813685248.2 4503599627370496.4 9007199254740993
```

Si l'on ne possédait pas une culture mathématique de base, on pourrait penser qu'à partir de 51, les puissances de 2 jouissent d'un *comportement chaotique*, imprévisible, car elles sont tantôt paires, tantôt impaires, parfois même non entières.

Si des puissances de 2 représentent la valeur d'un angle α exprimée en radians, itérées par doublement de l'angle, par exemple à partir d'un angle de 1 radian, de sorte que l'angle vaut 2 radians après la première itération, puis 4 radians après la seconde, puis 8 radians après la troisième, puis 16 radians après la quatrième, puis 32 radians après la cinquième, la valeur de l'angle est toujours prévisible : il vaut 2 puissance n radians après l'itération n.

Mais *prévisible* ne signifie pas *calculable*, ou, alors, il faut se doter des moyens adéquats, que l'on ne trouve pas, en général : si on veut conserver seulement 5 chiffres justes en base 10, c'est-à-dire entre 15 et 16 chiffres justes en base 2, il faudra disposer d'une arithmétique capable de calculer avec au moins 1016 chiffres en binaire, pour connaître SEULEMENT les 5 premiers chiffres justes de la valeur de 2 puissance 1000. Si l'on veut TOUS les chiffres justes, il faut une arithmétique représentant les nombres en binaire... sur **2000 bits**.

L'arithmétique à la norme IEEE, disponible sur PC, SUN, Macintosh et autres ordinateurs lambda, ne le permet pas.

Il faut se garder de croire qu'un ordinateur calcule juste et dispense son utilisateur de publier des résultats quelconques sans que ceux-ci soient accompagnés d'une estimation de l'erreur, relative ou absolue, maximale; une telle précaution évitera que l'on puisse tirer des conclusions fausses en interprétant des résultats calculés.

```
    2*64
1.8446744073709553E19
```

Et , pourtant, 2 puissance 64 se représente en binaire exactement par 1 suivi de 64 zéros.

L'ordinateur affiche un nombre en base 10, qu'il convient de multiplier par 10 puissance 19, mais ce nombre à multiplier ne possède que 17 chiffres significatifs connus (et encore, en restant optimiste).

Si ce nombre représente un angle en radians, et si l'on parvient à en prendre une ligne trigonométrique quelconque, par exemple le **sinus**, il ne faudra pas oublier les chiffres inconnus à droite, car ils représentent toujours des radians. Dans le cas présent, l'angle est connu avec une erreur de plus de 100 radians, et son sinus CALCULE SUR ORDINATEUR sera N'IMPORTE QUOI.

On pourrait en déduire, sinon, que le sinus d'un angle, possède un comportement chaotique... et ON l'a fait, et, hélas, publié maintes fois (du moins avec le carré du sinus).

En effet, si l'on prend un angle α QUELCONQUE, le carré de son sinus sera connu avec une précision d'environ 16 chiffres en base 10, initialement.

Il existe plusieurs manières de calculer le carré du sinus de l'angle 2α :

1) ON DOUBLE LA VALEUR DE α, ON PREND LE SINUS ET ON L'ELEVE AU CARRE.

Oui, mais... si on itère ce doublement, même en partant d'une valeur représentable comme 1 radian, on ne pourra pas obtenir une valeur suffisamment exacte, à partir de l'itération numéro 53 (l'erreur relative atteindra alors déjà 100%).

2) On utilise les formules trigonométriques pour transformer l'application :

sinus carré de 2α se récrit en fonction de α comme le carré de 2 $\sin\alpha \cos\alpha$

Posons alors **X**=$\sin^2\alpha$; alors, le carré du cosinus est 1−**X**, de sorte que l'application revient à itérer la formule $\mathbf{X}_{n+1}= 4\ \mathbf{X}_n\ (1-\mathbf{X}_n)$ c'est-à-dire l'***équation logistique*** bien connue. (Au passage, ce changement de variable prouve *sans appel* que ladite équation ne peut, en aucun cas, exhiber un comportement chaotique, puisque, inversement, son itération revient à itérer une application linéaire, le simple doublement de la variable α.

Mais, à chaque itération, l'ordinateur *introduit* une erreur statistique d'arrondi de 1/2 bit sur le dernier bit significatif de la représentation interne, ET *propage* l'erreur d'arrondi précédente, ceci toujours dans le même sens : l'erreur relative sur le carré d'une variable vaut **deux fois** l'erreur relative sur la variable - consulter les manuels de physique de la classe de seconde. L'erreur initiale double à chaque itération, puis, à la 52e itération, atteint le bit le plus significatif de la représentation interne du nombre (à la norme IEEE), donc le premier chiffre significatif de la représentation décimale (déjà malade à la 50e itération, car il faut 3 bits pour coder une puissance de 10).

3) On passe par le calcul matriciel.

On appelle une seule fois la fonction sinus, pour l'unique *condition initiale*, par exemple l'angle α de 1 radian.

On est certain que le nombre 1 est représentable en machine. On élève le sinus au carré (par multiplication avec lui-même), ce qui fournit **X** initial pour vérifier, le plus proprement possible, l'itération de l'équation logistique. Du sinus, on déduit le cosinus initial, de sorte que l'on forme la matrice de rotation :

M: $\cos\alpha$ $\sin\alpha$ dont le déterminant vaut, par définition 1, nombre $-\sin\alpha$ $\cos\alpha$ toujours codable dans tout ordinateur.

Si l'on multiplie cette matrice de rotation par elle-même, on obtient la matrice de rotation de l'angle 2α, par définition, dont le déterminant vaut 1, par définition aussi.

La valeur absolue du produit des termes de la deuxième diagonale donne alors le carré du sinus de l'angle courant, que l'on comparera à la valeur de **X** obtenue après la même itération, à chaque fois.

En ajoutant à cette valeur le produit des termes de la diagonale principale, on obtiendra le déterminant, dont on affichera la valeur à chaque itération.

L'avantage procuré par ce contrôle garde-fou provient du fait que l'on va vérifier un nombre que l'on sait mathématiquement égal à 1. Ainsi l'écart éventuel à 1 est l'**erreur absolue** sur le calcul, à chaque itération, mais aussi l'**erreur relative**. La vérification devient visuelle, et fort simple.

Travaux pratiques du cours numéro 1

a) Ecrire un programme en n'importe quel langage de programmation (évidemment en *double précision*) affichant le numéro de l'itération (0 pour la condition initiale), la valeur du carré du sinus de l'angle courant, la valeur de **X** itérée à l'aide de l'équation logistique, la valeur absolue du produit des termes de la seconde diagonale de la matrice de rotation associée, ainsi que la valeur courante du déterminant de cette dernière.

b) Faire tourner le programme aussi en *simple précision* et comparer les résultats.

c) (facultatif) Essayer en *quadruple précision*, ou bien avec *Mathematica*, (ou *Maple V*) en précision variable. Afficher, en couleur **RVB**, les graphes de la variation des trois résultats en fonction du numéro de l'itération, et en tenant compte de l'erreur relative fournie par le calcul du déterminant : il suffit d'afficher la barre d'erreur comme une ligne verticale colorée, pour chaque paramètre. Alors, par superposition des trois couleurs, la zone (ex-chaotique) pour laquelle il est impossible de tirer des conclusions lorsqu'on a travaillé avec un ordinateur, apparaîtra, visuellement, en blanc cassé ou en blanc intense, selon que l'on aura géré les couleurs en intensité normale ou en surintensité.

COURS Numéro 2. Essais comparatifs.

Faire tourner le programme de la leçon numéro 1 sur des ordinateurs de marques **différentes** : on suggère le PC (IBM ou clones) et Macintosh (APPLE), car les micro-processeurs de ces deux machines sont de deux marques **différentes** mais respectent une même norme de codage des constantes en virgule flottante, la norme IEEE, effectivement avec 64 bits pour les constantes en double précision, dont 52 bits pour la partie signifiante, la mantisse (les autres codant l'exposant et les champs du signe).

Comparer les résultats obtenus avec un **même** programme (on aura intérêt à choisir un langage de programmation normalisé), avec la même valeur initiale de α (par exemple 1 radian) et avec la même norme de codage de l'information (IEEE), de sorte que l'on puisse dire si oui ou non, à partir des **MEMES** conditions initiales, il est possible de parvenir à des résultats **différents**.

Réfléchir ensuite au fait que les constructeurs de ces deux machines ont failli nous empêcher de pouvoir effectuer ce test, car ils s'étaient mis d'accord pour mettre sur le marché des machines équipées du **MEME** micro-processeur, le *Pentium (Bogué?).*

Nous l'avons échappé belle. Un peu plus, et le chaos s'engouffrait dans la brèche.

[1] On utilise pour cela une symbolique normalisée, à la fois notation, langage de commande et de programmation vectorielle
(ISO8485, Genève, 1989, et AFNOR); à votre avis, laquelle ?