TD: calculs en virgules flottantes, erreurs numériques

(*) très facile (1-4 lignes), (**) moyen, (***) des subtilités

Exercice 1 : Erreurs liées aux calculs en virgule flottante

- (*) Soit $\mathfrak{u} \approx 1.11e-16$ l'unité d'arrondi de votre python (ordinateur avec registres 64 bits) (i.e., $1 \oplus 2e-16 > 1$, $1 \oplus 1e-16 = 1$). Soient $\oplus, \ominus, \otimes, \oslash$ les opérations arithmétiques numériques usuelles. Quels sont les calculs sans erreur et avec erreur ?

$$0,625 \oplus 0,3125e16 =$$
$$0,625 \ominus 0,3125e16 =$$
$$0,625 \otimes 0,3125e16 =$$
$$0,625 \oslash 0,3125e16 =$$

- (*) On propose d'approcher le calcul de $\ln n$ par la formule itérative : $F_n := -\gamma + \sum_{k=1}^{n} \frac{1}{k}$, avec $\gamma \approx 0,577215$ (constante d'Euler). Théoriquement l'erreur est proche de $\frac{1}{n}$.

  Combien faut-il d'itérations pour estimer $\ln n$, pour n entier ? Programmez le calcul en python. Combien de temps faut-il pour estimer $\ln 1e6$, $\ln 1e9$, $\ln 1e12$?

  Mathématiquement la suite F_n tend vers l'infini. Pourquoi ? Mais numériquement, la suite F_n est convergente ! Pourquoi ? A partir de quel n est-elle constante environ (sous python, vous avez à peu près 16 décimales correctes, i.e. $\mathfrak{u} \approx 1e-16$).

- (**) On propose maintenant de calculer $\ln x$ à partir de $\exp x$ avec une approche dichotomique. On supposera pour simplifier que x est entre -500 et 500. Proposez un algorithme récursif pour faire ce calcul. Combien d'itérations faut-il pour que l'algorithme retourne une précision de 10 décimales après la virgule ? Est-ce plus rapide que l'approche précédente ? Quel est le défaut de cette algorithme néanmoins ?

- (*) On vous donne une liste de valeurs L entre 0 et 1. Quel algorithme donnera la somme avec le plus de précision ? La somme des éléments dans l'ordre où ils sont stockés ? la somme des éléments dans l'ordre croissant ? la somme des éléments dans l'ordre décroissant ? Si on avait des valeurs négatives, quel est le bon ordre ?

- (**) Soit la suite $x_0 = 1, x_1 = \frac{1}{6}, x_{n+1} = \frac{37}{6}x_n - x_{n-1}$ pour tout $n \geq 1$. Vérifiez que $x_n = \frac{1}{6^n}$. Faites un programme python qui calcule les 20 premiers termes de la suite. Que constatez-vous ?

  Calculez le conditionnement de la fonction $f(x, y) = \frac{37}{6}x - y$ au voisinage de $(1/6^n, 1/6^{n-1})$. Expliquez ensuite pourquoi l'erreur grandit environ de 37 fois à chaque étape.

- (*) Calculez $f(x) = \frac{1}{1-\sqrt{1-x^2}}$ au voisinage de 0 à l'aide de python ou matlab (par exemple testez $0,0001$, $0,00001$, $0,000001$, $0,0000001$, $0,00000001$). Que constatez-vous ? Où se situe le problème ?

  Quelles seraient les valeurs attendues au voisinage de 0 ?

  Proposez une autre écriture de la même fonction, qui retourne des valeurs beaucoup plus précises au voisinage de 0.

Take-away message

- L'unité d'arrondi vous donne la précision relative maximum que vous pouvez espérer dans un calcul numérique, soit environ 16 décimales significatives avec 64 bits.
- L'ordre dans lequel on fait les calculs a une importance dans la qualité du résultat. Groupez les calculs entre valeurs de même ordre de grandeur.
- Il peut être intéressant de travailler la formule pour son calcul soit plus précis: on essaie d'éviter autant que possible les domaines de calcul où le conditionnement est grand.