

Algorithmique numérique : Examen Janvier 2021-2022

Retranscrit par Jacques Hogge, correction par Doeraene Anthony

February 5, 2022

1. Question 1

Etant donné le nombre 28.375

a) Quelle est la représentation binaire à points fixe du nombre avec 8bits(k=8) pour la partie fractionnaire. le format attendu est XXXXXXXX.XXXXXXXX avec X=0 ou 1.

0001 1100.0110 0000

b) Quelle serait la valeur du bit31 (1bit) en utilisant une représentation float32

0 car positif

c) Quelle serait la valeur des bit22 à 14 (8bits) en utilisant une représentation float32

1100 0110 car bits de mantisse prend représentation en float de question a = 11100.011 et on shift jusqu'à n'avoir plus qu'un bit : 1.1100011 => la mantisse est donc 11000110 (en comblant le dernier bit)

d) Quelle serait la valeur des bit30 à 23 (8bits) en utilisant une représentation float32

1000 0011 : à partir de c, on a du shifter de 4 à droite : doit faire donc $\times 2^4$. La valeur de l'exposant est donc $x - 127 = 4 \Leftrightarrow x = 131$ et $131 = 128 + 2 + 1$

e) Dans ce cas particulier, quelle est l'erreur absolue sur cette représentation float32

0 : on représente exactement le nombre

2. Question 2

Etant donné la fonction suivante avec x,y et z des nombres positif et avec x et y ayant généralement des valeurs proches:

```
def f(x,y,z): return np.log(np.exp(x) np.exp(-y) np.exp(z))
```

a) Améliore la stabilité/ qualité numérique du code

def f(x,y,z) : return x+z-y ; (doit éviter la soustraction entre deux nombres proches)

b) Quelle est l'erreur relative maximale liée au arrondi sur l'opérateur xy en float32

$$e_f = y * e_x + x * e_y$$

$$e_f = x * y * \epsilon_x + x * y * \epsilon_y$$

$$e_f = x * y * (\epsilon_x + \epsilon_y)$$

$$\epsilon_f = (\epsilon_x + \epsilon_y)$$

L'erreur maximale est donc, en notant $\epsilon_x = \epsilon_y = 2^{-p}$ et $p = 24$ car travaillant en float32 : $2^{-24} + 2^{-24} = 2^{-23}$ en rajoutant l'erreur de float on arrive au même résultat ($3 * 2^{-24}$ possible aussi selon calcul des floats)

3. Question 3

La vitesse d'une fusée est mesurée a trois moment différents : (t_1, t_2, t_3) . Nous voulons interpoler par un polynôme la vitesse de la fusée au temps t en utilisant ces trois mesures (v_1, v_2, v_3) . Ecris le polynôme de Lagrange $P(t)$. La solution doit avoir une forme algébrique (pas numérique) et ne faire intervenir que les variables : $t, t_1, t_2, t_3, v_1, v_2, v_3$

$$P(t) = \frac{(t - t_2)(t - t_3)}{(t_1 - t_2)(t_1 - t_3)} * v_1 + \frac{(t - t_1)(t - t_3)}{(t_2 - t_1)(t_2 - t_3)} * v_2 + \frac{(t - t_1)(t - t_2)}{(t_3 - t_1)(t_3 - t_2)} * v_3$$

4. Question 4

Si les mesures de vitesse de la fusée de la question 3 sont celles-ci-dessous, quel était la vitesse de la fusée a $t=2.5$

t	1.0	2.0	3.0
v(t)	1.0	2.0	5.0

Marche a suivre : Utilise une interpolation cubique-spline "naturelle"

a) quelle est la matrice (tri-diagonal) associée au système des courbures

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 4 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

b) quelle sont les courbures de la cubic-spline

cubiques-spline naturelle donc $k_1 = k_3 = 0$. En appliquant ensuite la formule, $k_1 + 4 * k_2 + k_3 = 12$ et donc $k_2 = 3$

c) quelle est la vitesse de la fusée au moment $t=2.5$

Via la formule en remplaçant les valeurs : 3.3125

5. Question 5

Complètes les bouts de code manquant pour rendre opérationnel cet algo qui localise les racines de la fonction f par la méthode de bisection avec une précision supérieure ou égale au paramètre 'tol'. La racine de f se trouve a priori entre x_1 et x_2

```
import math
from numpy import sign
def bisection(f, x1, x2, tol=1.0e-9):
    f1 = f(x1)
    f2 = f(x2)
    if f1 == 0.0 : return x1
    if f2 == 0.0 : return x2
    if sign(f1) == sign(f2):
        raise Exception('Root is not bracketed')
    n = # a) compléter le code manquant

    for i in range(n):
        x3 = # b) compléter le code manquant
        f3 = f(x3)
        if f3 == 0.0 : return x3
        if sign(f2) != sign(f3):
```

```

    # c) completer le code manquant
else :
    # d) completer le code manquant
return (x1, x2)/2.0

```

- a) `int(math.ceil(np.log(abs(x2 - x1)/tol) / np.log(2)))`
- b) `(x1+x2)/2 ; f3 = f(x3)`
- c) `x1 = x3; f1 = f3`
- d) `x2 = x3; f2 = f3`

6. Question 6

Dérive l'approximation par différence finie vers l'avant de deuxième ordre (second order forward finite difference approximation) de $f(x)$ à partir des expansions de Taylor de f autour de x

- a) Quelles sont les deux expansions de Taylor nécessaires à ce développement

$$f(x+h) = f(x) + h * f'(x) + h^2 * \frac{f^{(2)}(x)}{2!} + O(h^3)$$

$$f(x+2h) = f(x) + 2h * f'(x) + 4 * h^2 * \frac{f^{(2)}(x)}{2!} + O(h^3)$$

- b) quelle est l'approximation de $f'(x)$ dans ce cas

En éliminant $f^{(2)}$ en faisant $f(x+2h) - 4f(x+h)$ et en isolant $f'(x)$

$$f'(x) = \frac{-3 * f(x) + 4 * f(x+h) - f(x+2h)}{2h}$$

- c) quel est l'ordre de l'erreur sur cette approximation (en terme de h)

$O(h^2)$ Car forward second en voyant l'ordre restant après avoir isolé $f'(x)$

7. Question 7

Etant données les données suivantes:

x	3.0	4.0	5.0	6.0	7.0
f(x)	1.2	2.3	3.5	4.7	5.9

- a) Estime $f'(x)$ en $x=3.0$ en utilisant la formule dérivée à la question précédente

En remplaçant dans la formule avec $h = 1$ on obtient $f'(3) = 1.05$ (ou en utilisant $h = 2$, on obtient $f'(3) = 1.125$)

- b) Estime $f'(x)$ en $x=3.0$ en utilisant la méthode la plus précise possible

En utilisant Richardson avec les deux approximations précédentes, on trouve $f'(3) = \frac{4*1.05-1.125}{3} = 1.025$

- c) Dans le deuxième cas, quel est l'ordre de l'erreur sur l'approximation (en terme de h)

$O(h^4)$

8. Question 8

Etant données les données de la question 7:

a) Quelle est l'intégrale de la fonction $f(x)$ entre 3 et 7 en utilisant Newton-Cotes avec $n=1$ (et la méthode composite si possible)

$$I = \frac{h}{2} * (f(3) + 2f(4) + 2f(5) + 2f(6) + f(7))$$

$$I = \frac{1}{2} * (28.1)$$

$$I = 14.05$$

b) Quel est l'ordre de l'erreur commise sur cette intégral en terme de h
 $O(h^2)$

c) quelle méthode pourrait-on utiliser pour réduire l'erreur et quel serait alors l'ordre de l'erreur réduite

Romberg avec 5 points = 4 panels. On peut déterminer k en posant via les formules $2^{k-1} = \text{panels}$, on trouve donc $k = 3$ et l'ordre de l'erreur est donc $O(h^{2*k}) = O(h^6)$

9. Question 9

Nous souhaitons résoudre le problème de Cauchy (Initial Value Problem) suivant $y'' = -y - 0.1y'$ avec les conditions initiales $y(0)=0$ et $y'(0)=1$. Nous pouvons utiliser la méthode RK4 qui a comme paramètre F , $xstart$, $xstop$, $ystart$ et h . Ecris la fonction $F(y,x)$ que nous devons utiliser pour résoudre ce problème

```
def F(y, x):
    return np.array([ y[1],
                      -y[0] - 0.1*y[1]])
```

10. Question 10

Nous voulons optimiser les variables de design x, y afin de minimiser le coût associé.

Le cout est proportionnel à $y + (y - 2)^2 + 5yx$

En plus nous devons respecter les contraintes suivantes :

- (a) $x \geq 0$
- (b) $1 \leq y \leq 10$
- (c) $x + y = 10$

a) Ecris la fonction objective que nous devons passer à l'algo Powell pour résoudre ce problème. la variable $\text{Lambda} = 1$ (définie globalement) pour être utilisée pour modérer l'impact d'une éventuelle pénalité

```
def f(x, y):
    return y + (y-2.0)**2 + 5*x*y + Lambda*(min(x, 0)**2
    + max(y-10, 0)**2 + min(y-1, 0)**2 + (x+y-10)**2)
```

b) Quelle procédure adopter si les variables de design optimal violent les contraintes ?

Augmenter la valeur de Lambda et recommencer l'optimisation à partir du précédent point optimal trouvé