TP1: Erreurs Numériques

September 8, 2016

1 Utilisation de base de Python

Exercice. On veut tracer les polynômes $P(x) = ax^2+bx+c$ et $Q(x) = cx^2+bx+a$ pour x compris entre 0 et 2.

Définir les fonctions P et Q pour $a = 10$, $b = -15$ et $c = 5$.

1) Créez un vecteur X allant de 0 à 2 avec un pas de h (que vous pourrez varier au besoin).

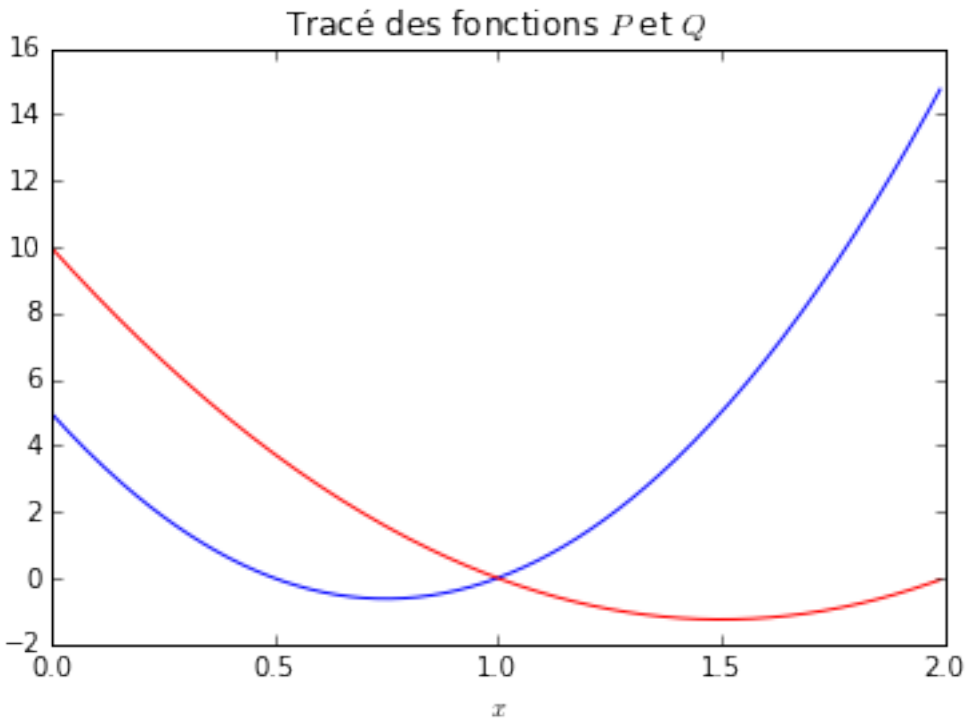
2) Créez deux vecteurs Y_1 et Y_2 contenant les images par P et Q du vecteur X.

3) Tracer les graphes de Y_1 et Y_2 en fonction de X. Mettre un titre, les axes, légendes,..

```
In [1]: def P(x,a,b,c): # définition du polynôme P
            return a*x**2+b*x+c
        def Q(x,a,b,c): # définition du polynôme Q
            return c*x**2+b*x+a
```

```
In [ ]:
```

```
In [2]: #Représentation des polynômes P et Q
        import numpy as np # importation de la librairie NUMPY
        import matplotlib.pyplot as plt # importation de la librairie MATPLOT LIB
        h=1./100 # définition du pas de discrétisation
        a,b,c=10., -15. , 5.
        X=np.arange(0,2,h) #définition de la discrétisation spatiale
        Y1=P(X,a,b,c)
        Y2=Q(X,a,b,c)
        plt.plot(X,Y1) #tracé de la fonction P, couleur bleue par défaut
        plt.plot(X,Y2,'r') #tracé de la fonction Q, en rouge
        plt.xlabel(u'$x$')
        plt.title(u'Tracé des fonctions $P$ et $Q$')
        plt.show()
```



2 Erreur et Stabilité

2.1 Représentation des nombres

2.1.1 Représentation des nombres en base 10

Pour tout $x \in \mathbb{R}$, il existe $N \in \mathbb{Z}$ tel que $10^N \leq x < 10^{N+1}$ et une suite $(a_i)_{i\in\mathbb{N}}$ tel que $x = \sum_{i=0}^{\infty} a_i \, 10^{N-i}$.

Exercice: écrire $x = \dfrac{3}{8}$ et $x = \dfrac{4}{3}$ en base 10.

Attention certains nombres peuvent avoir deux représentations: par exemple

$$0.9999.... = \sum_{n=1}^{\infty} \frac{9}{10^n} = 1.$$

2.1.2 Représentation des nombres en base 2

Cette base est particulièrement intéressante en informatique car pour transmettre ou stocker des nombres, il suffit d'un mécanisme physique à deux états. Pour tout $x \in \mathbb{R}$, il existe $N \in \mathbb{N}$ tel que $2^N \leq x < 2^{N+1}$ et il existe une suite $(a_i)_{i\in\mathbb{N}}$ tel que $a_i \in \{0, 1\}, \forall i \in \mathbb{N}$ et $x = \displaystyle\sum_{i=0}^{\infty} a_i \, 2^{N-i}$.

Exemple 1: en base 2, $x = 10 = 1 \times 2^3 + 1 \times 2^1$ s'écrit $x = 1010$.

Exercice: écrire en base 2 les nombres $x_1 = \dfrac{3}{8}$, $x_2 = \dfrac{1}{10}$, $x_3 = \dfrac{1}{3}$.

2.1.3 Virgule flottante en notation scientifique

Les systèmes précédents sont à virgule fixe, la virgule séparant partie entière et partie "décimale". En base 10, les nombres 12.345, 0.12345×10^2 et 12345×10^{-3} sont tous égaux. Ils sont tous de la forme $\pm m \times b^e$:

- m s'appelle la mantisse,
- b désigne la base,
- e s'appelle l'exposant.

Pour éviter les redondances et fixer les notations, on introduit la notion de virgule flottante: elle consiste à choisir l'exposant e de telle sorte que la mantisse s'écrive

$$m = \alpha_0.\alpha_1\,\alpha_2\,...\alpha_p\,..., \quad \alpha_0 > 0.$$

2.2 Représentation des nombres en machine

Dans la suite, on considère l'ensemble des nombres réels en base 2. Il est impossible de tous les représenter en machine car cet ensemble est infini. Pour les nombres entiers, ils sont stockés dans des éléments de mémoire formé de cellules constituées d'unités élémentaires appelées *bits*. Une cellule est constituée en général de 8 bits et appelée *octet*.

- Si un ordinateur utilise 2 octets, il représentera les entiers entre -2^{15} et $2^{15}-1$. En effet, on représentera les entiers sous la forme

$$x = -\alpha_{15} \times 2^{15} + \sum_{j=0}^{14} \alpha_j \times 2^j.$$

- Si un ordinateur utilise 3 octets, il représentera les entiers entre -2^{23} et $2^{23} - 1$.

Pour les réels, on ne peut représenter qu'un sous ensemble fini, appelé ensemble des réels flottants, noté $\mathcal{F}$. Cet ensemble dépend bien évidemment de la machine et de la base utilisée. En base 2, l'ensemble $\mathcal{F}$ est donné par

$$\mathcal{F} = \{\pm m \times 2^e, \quad m = a_1.a_2...a_p, \quad a_i \in \{0,1\}, \quad e \in [e_{min}, e_{max}]\}$$

Le nombre p de chiffres dans la mantisse est le nombre de chiffres significatifs. Les deux principaux formats (standards IEEE) sont

- Précision simple (32 bits): $23 + 1$ (signe) bits affectés à la mantisse m et 8 bits affectés à l'exposant donc $e \in [-128, 127]$
- Précision double (64 bits): $52 + 1$ (signe) bits affectés à la mantisse m et 11 bits affectés à l'exposant donc $e \in [-1024, 1023]$.

Le plus grand nombre représentable en valeur absolue est $x_{max} = a_1.a_2...a_p \times b^{e_{max}}$ avec $a_i = 1$. Soit $x_{max} = (2 - 2^{1-p}) \times 2^{e_{max}}$. Au dela, on ne peut représenter un nombre en machine, c'est le seuil d'*overflow*.

Pour la valeur minimale en valeur absolue x_{min}, cela devrait être $x_{min} = 2^{e_{min}}$ si on conserve la normalisation. En deça de cette valeur, on peut *dénormaliser* et admettre $a_1 = 0$ dans la mantisse. Dans ce cas, $x_{min} = 2^{e_{min}-p+1}$.

Une fois $\mathcal{F}$ défini, il faut créer une application d'arrondi, qu'on notera $fl = [-x_{max}, x_{max}] \to \mathcal{F}$. Deux stratégies sont possibles:

- Arrondi par troncature: $fl(x) = \max\{y \in \mathcal{F},\ y \leq x\}$ si $x > 0$ et $fl(x) = \min\{y \in \mathcal{F},\ y \geq x\}$ si x<0.
- Arrondi au plus près: $\forall y \in \mathcal{F}, |x - fl(x)| \leq |x - y|$.

2.2.1 Opérations arithmétiques élémentaires

L'ensemble $\mathcal{F}$ n'est pas stable. Ainsi l'opération $x+y$ devient $fl(fl(x)+fl(y))$ et les erreurs d'arrondis peuvent s'accumuler. L'exemple le plus notable est le *mécanisme d'absorption*. Soit $x = m_x \times 2^{e_x}$ et $y = m_y \times 2^{e_y}$. Supposons $e_x > e_y$ et notons $n = e_x - e_y$. On a donc

$$fl(x) + fl(y) = (m_x + 2^{-n} \times m_y) \times 2^{e_x}.$$

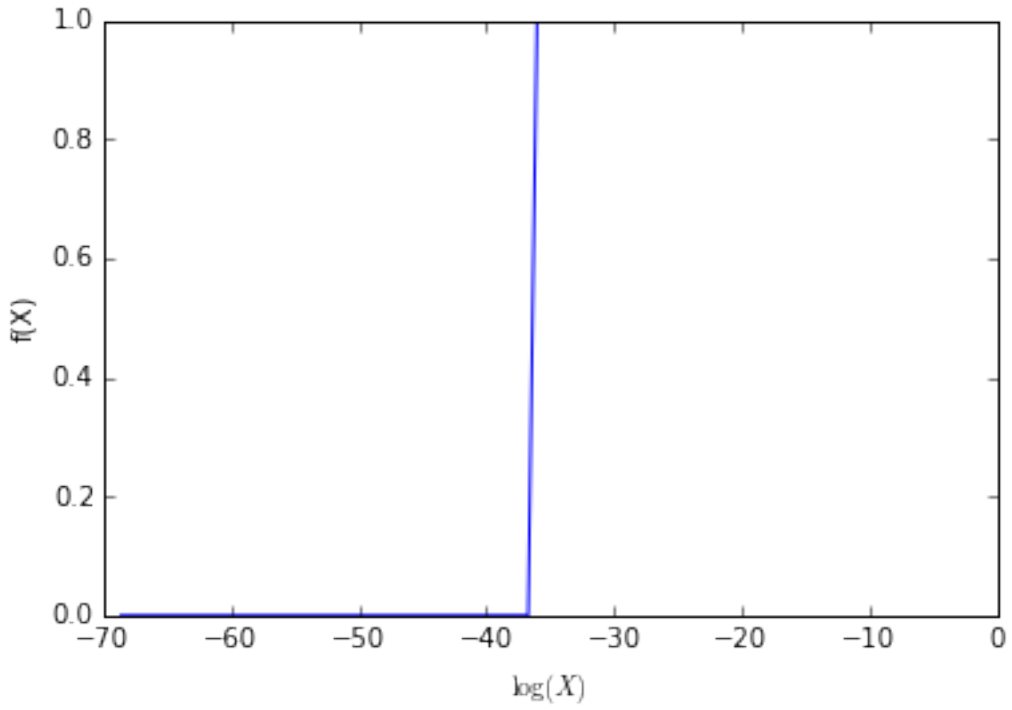
Le nombre $m_x + 2^{-n} \times m_y$ possède plus de p chiffres et donc une partie des chiffres de m_y est perdue.Si n est proche de p, alors on perd tous les chiffres de la mantisse et on obtient alors:

$$fl(fl(x) + fl(y)) = fl(x).$$

3 Exemples d'erreurs numériques

3.1 Exemple 1: calcul de $f(x) = \dfrac{(1+x)-1}{x}$ pour x proche de 0.

```
In [1]: import math as mt # importation de la librairie MATH
        import numpy as np # importation de la librairie NUMPY
        import matplotlib.pyplot as plt # importation de la librairie MATPLOT LIB
        def f(x): #definition de la fonction f
            return ((1+x)-1)/x
        a=0.5
        N=np.arange(1.,100.) #Création du vecteur N=[1:100] avec un pas de 1
        X=a**N #Création du vecteur a^N, pour N allant de 1 à 100
        plt.plot(np.log(X),f(X))
        plt.xlabel(u'$\log(X)$')
        plt.ylabel(u'f(X)')
        plt.show()
```

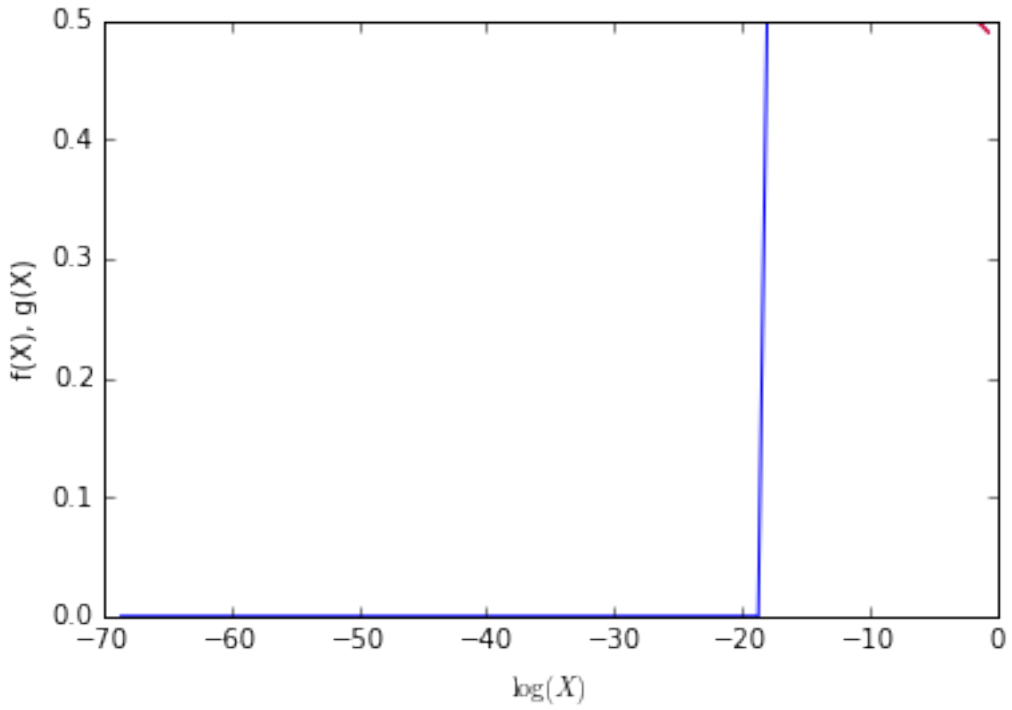


3.2 Exemple 2: Calcul de $\lim_{x\to 0} \dfrac{1-\cos(x)}{x^2}$.

Travail à réaliser

- Montrer que $1-\cos(x) = 2\sin^2(\frac{x}{2})$. En déduire $\lim_{x\to 0} \dfrac{1-\cos(x)}{x^2}$.
- Créer un vecteur n allant de 0 à 100 avec un pas de 1.
- Créer un vecteur X contenant les valeurs $(1/2)^k$ avec $k \in \{0, \cdots, 100\}$.
- Définir $f : x \mapsto \dfrac{1-\cos(x)}{x^2}$ et $g : x \mapsto \dfrac{1}{2}\left(\dfrac{\sin(x/2)}{x/2}\right)^2$.
- Représenter f et g. Quelle est la formulation la plus adéquate pour étudier la limite de f en $x = 0$. Expliquer.

```
In [2]: import math as mt # importation de la librairie MATH
        import numpy as np # importation de la librairie NUMPY
        import matplotlib.pyplot as plt # importation de la librairie MATPLOT LIB
        def f(x): #definition de la fonction f
            return (1-np.cos(x))/x**2
        def g(x): #definition de la fonction g
            return 0.5*(np.sin(x/2)/(x/2))**2
        a=0.5
        N=np.arange(1.,100.) #Création du vecteur N=[1:100] avec un pas de 1
        X=a**N #Création du vecteur a^N, pour N allant de 1 à 100
        plt.plot(np.log(X),f(X)) #tracer de f
        plt.plot(np.log(X),g(X),'r') #tracer de g
        plt.xlabel(u'$\log(X)$')
        plt.ylabel(u'f(X), g(X)')
        plt.show()
```



3.3 Exemple 3: Calcul d'une dérivée

Soit $f : x \mapsto \sin(x)$. On souhaite calculer numériquement la dérivée de f en $x = 1$ à l'aide de la formule

$$f'(1) = \lim_{h \to 0} \frac{f(1+h) - f(1)}{h}.$$

Travail à réaliser

- Créer un vecteur H contenant les valeurs 10^{-k} pour $k \in \{0, \ldots, 16\}$ avec un pas de 1.
- Pour tout $h \in \mathbb{R}$, on note $e(h) = \left| f'(1) - \frac{f(1+h) - f(1)}{h} \right|$.Représenter $e(H)$: pour quelle valeur de h a-t-on la meilleure approximation de $f'(1)$?
- Sachant que la fonction sin est fournie avec une erreur de l'ordre de la précision machine, not'ee u, montrer que

  $$|e(h)| \leq \frac{h}{2} \sup_{\xi \in [1, 1+h]} |f''(\xi)| + \frac{2\,u}{h}.$$

  Pour quelle valeur de h la fonction e est minimale?
- Expliquer le phénomène observé et donner un ordre de grandeur de u.
- Refaire l'exercice en utilisant la formule $f'(1) = \lim_{h \to 0} \frac{f(1+h) - f(1-h)}{2h}$.

```
In [6]: import math as mt # importation de la librairie MATH
        import numpy as np # importation de la librairie NUMPY
        import matplotlib.pyplot as plt # importation de la librairie MATPLOT LIB
```

```
def e(x): #definition de la fonction erreur e
    return np.abs(np.cos(1)-(np.sin(1+x)-np.sin(1))/x)
def e2(x): #definition de la fonction erreur e avec une formule plus précise
    return np.abs(np.cos(1)-(np.sin(1+x)-np.sin(1-x))/x)
a=1./10
N=np.arange(1.,16.)
H=a**N #Création du vecteur 10^-k, pour k allant de 1 à 16.
plt.plot(np.log(H),np.log(e(H))) #tracer de l'erreur avec la première formule
# plt.plot(np.log(X),g(X),'r') #tracer de l'erreur avec la deuxième formule
plt.xlabel(u'$\log(H)$')
plt.ylabel(u'$\log\, e(H)$')
plt.show()
```

3.4 Stabilité d'un algorithme

Cette partie vise à montrer l'instabilité associée à certaines relations de récurrence. Les intégrales $I_k = \int_0^1 x^k \exp(-x)dx$, avec $k \in \mathbb{N}$, peuvent être calculées selon la relation de récurrence suivante:

$$I_0 = 1 - \frac{1}{e}, \qquad I_{k+1} = (k+1)\, I_k - \frac{1}{e}.$$

Travail à réaliser

- Utiliser la relation de récurrence pour calculer les valeurs I_0 à I_{30}. Créer le vecteur I_c qui contiendra les valeurs I_k pour k allant de 0 à 30.

La relation de récurrence peut aussi s'écrire $I_k = \dfrac{I_{k+1} + \dfrac{1}{e}}{k+1}$.

- Utiliser ce deuxième algorithme pour calculer les valeurs I_{30} à I_0 en choisissant la valeur initiale $I_{30} = 1$. Stocker les valeurs obtenues dans le vecteur I_d.
- Faire un graphe représentant I_c et I_d en fonction du nombre d'itérations pour visualiser les résultats.

In [20]:
```
import math as mt
import numpy as np
import matplotlib.pyplot as plt
I=np.zeros(50)
# Méthode 1 instable
I[0]=1-mt.exp(-1.)
for j in range(len(I)-1):
    I[j+1]=(j+1)*I[j]-mt.exp(-1.)
J=np.zeros(50)
# Méthode 2 stable
J[len(J)-1]=(1+mt.exp(-1))/102
for j in reversed(range(len(J)-1)):
    J[j]=(J[j+1]+mt.exp(-1))/(j+1)

plt.plot(range(len(J)),J)
# plt.plot(range(len(I)),I,'r')
plt.show()
print I
print J
```

```
[  6.32120559e-01   2.64241118e-01   1.60602794e-01   1.13928941e-01
   8.78363239e-02   7.13021781e-02   5.99336275e-02   5.16559512e-02
```

```
   4.53681687e-02   4.04340776e-02   3.64613345e-02   3.31952383e-02
   3.04634190e-02   2.81450054e-02   2.61506350e-02   2.43800845e-02
   2.22019104e-02   9.55303570e-03  -1.95924799e-01  -4.09045061e+00
  -8.21768917e+01  -1.72608261e+03  -3.79741852e+04  -8.73406628e+05
  -2.09617594e+07  -5.24043986e+08  -1.36251436e+10  -3.67878878e+11
  -1.03006086e+13  -2.98717649e+14  -8.96152948e+15  -2.77807414e+17
  -8.88983724e+18  -2.93364629e+20  -9.97439738e+21  -3.49103908e+23
  -1.25677407e+25  -4.65006406e+26  -1.76702434e+28  -6.89139494e+29
  -2.75655797e+31  -1.13018877e+33  -4.74679283e+34  -2.04112092e+36
  -8.98093204e+37  -4.04141942e+39  -1.85905293e+41  -8.73754878e+42
  -4.19402341e+44  -2.05507147e+46]
[ 0.63212056  0.26424112  0.16060279  0.11392894  0.08783632  0.07130218
  0.05993363  0.05165595  0.04536817  0.04043408  0.03646133  0.03319524
  0.03046344  0.02814522  0.02615358  0.02442426  0.02290878  0.02156988
  0.02037847  0.0193115   0.01835047  0.01748038  0.01668893  0.01596593
  0.01530288  0.01469264  0.01412914  0.01360721  0.01312243  0.01267096
  0.0122495   0.01185515  0.01148538  0.01113795  0.01081091  0.01050252
  0.01021122  0.00993563  0.00967452  0.00942677  0.00919139  0.00896747
  0.00875419  0.00855082  0.00835668  0.00817116  0.00799374  0.00782627
  0.00778143  0.01341058]
```

4 Exercices de synthèse

Exercice 1

Donner un programme permettant de calculer correctement les fonctions suivantes au voisinage de $x = 0$ (on commencera par en donner un équivalent):

$$f(x) = \sqrt{1+x} - 1, \qquad g(x) = \frac{1-\cos(x)}{\sin(x)}.$$

```
In [ ]: ecrire le code
```

Exercice 2

On souhaite calculer les intégrales

$$\forall n \in \mathbb{N}, \quad 1 \leq n \leq 50, \quad I_n = \int_0^1 \frac{x^n}{x^2 + 9\,x + 20} dx.$$

- Montrer que les I_n vérifient une relation de récurrence $I_{n+1} = a_n\, I_n + b_n\, I_{n-1} + c_n$ où a_n, b_n, c_n ne dépendent que de n.
- En supposant I_0, I_1 donnés avec une erreur ε, déterminer l'erreur commise e_n sur le calcul de I_n en utilisant la formule de récurrence.
- Montrer que $\dfrac{1}{30(n+1)} \leq I_n \leq \dfrac{1}{20(n+1)}$.
- Proposer et programmer une méthode pour calculer I_{10}.

```
In [ ]: ecrire le code
```