Séance 2: La méthode de Newton

L1 – Université Nice Sophia Antipolis

Objectifs:

- Introduction aux flottants
- Savoir mesurer le temps de calcul d'une fonction
- Mettre en oeuvre la méthode de Newton
- Notion de graphe de contrôle

Exercice 1 (Récursion versus boucle while, $\star$)

1. *Programmez la fonction prenant un entier $n \geq 0$ et retournant la factorielle n ! de n :*
   (a) *par récurrence, sous la forme d'une fonction* `fac_rec(n)` *Testez sur* `fac_rec(5)` *qui vaut 120 ;*
   (b) *en utilisant une boucle while, sous la forme d'une fonction* `fac(n)`*. Testez sur* `fac(5)`*.*
2. *Python est-il capable*[1] *de calculer $1000!$ avec chacune de ces définitions ?*
3. *Pour chronométrer un calcul en Python, il suffit d'importer la fonction* `time()` *du module* `time`*. Le résultat de* `time()` *est un nombre en secondes flottantes depuis une date arbitraire, seule compte la différence entre deux appels à* `time()` *pour mesurer une durée écoulée. Exemple :*

```
from time import time # importation de la fonction time()
start = time() # top chrono !
f = fac(1000) # le calcul proprement dit
end = time() # stop chrono !

# affichage du résultat
print('f=',f)
print('Time =',end-start,'s')
```

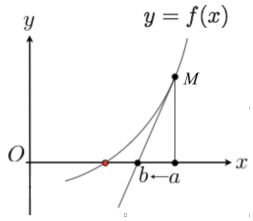
   Qui est le plus rapide entre `fac_rec(1000)` *et* `fac(1000)` *?*

□

1. On peut faire calculer `fac(5000)` par récurrence en Python en utilisant la fonction `sys.setrecursionlimit(...)` mais il n'est pas toujours clair de trouver la bonne limite...

Exercice 2 (Méthode de Newton, ⋆⋆)

Les calculatrices possèdent souvent une touche Solve permettant de calculer une racine d'une équation $f(x) = 0$. Par exemple, il est difficile sans machine de trouver une solution réelle à l'équation $x^5 - 3x + 1 = 0$. Nous allons faire abstraction de la fonction f, la supposer dérivable et à dérivée non nulle presque partout (de sorte que la tangente à la courbe existe avec une probabilité quasi-nulle d'être horizontale) pour appliquer la méthode des tangentes de Newton vue en cours.



On suppose que l'approximation courante est $a > 0$. L'équation de de la droite tangente à la courbe de f au point $M(a, f(a))$ s'écrit $y - f(a) = f'(a)(x - a)$: l'approximation améliorée b vaut donc

$$b = a - \frac{f(a)}{f'(a)}.$$

Pour $f(x) = x^2 - r$, on a $f'(x) = 2x$ et $b = a - \frac{a^2-r}{2a} = \frac{1}{2}(a + \frac{r}{a})$ comme vu en cours.

1. *Écrivez une fonction* `deriv(f,a,h)` *qui renvoie $\frac{f(a+h)-f(a)}{h}$: pour un h petit, c'est une bonne approximation de $f'(a)$*
2. *Écrivez une fonction* `solve(f,a,h)` *qui renvoie un nombre x tel que $|f(x)| < h$ en prenant pour approximation initiale a.*
3. *Testez votre fontion : calculez les cinq premières décimales de $\sqrt{2}$; vous devez trouver $1.41421\ldots$*

□

Exercice 3 (Fermat, Grothendieck, et les nombres premiers, ⋆⋆)

Le n-ième nombre de Fermat est $2^{2^n} + 1$. Fermat avait conjecturé que tous ces nombres étaient des nombres premiers. Votre mission : montrer à l'aide de Python que cette conjecture est fausse !

1. *Écrivez une fonction* `fermat(n)` *qui renvoie le n-ième nombre de Fermat. Par exemple,* `fermat(3)` *renvoie 257.*
2. *Écrivez une fonction* `premier_facteur(n)` *qui renvoie le plus petit nombre ≥ 2 qui divise n. Par exemple,* `premier_facteur(35)` *renvoie 5 et* `premier_facteur(31)` *renvoie 31.*
3. *Que pensez-vous de 57, appelé nombre premier de Grothendieck ? Testez de même la primalité de* `fermat(3)` *et* `fermat(4)` *en utilisant la fonction* `premier_facteur`*.*
4. *Écrivez un programme qui affiche le premier n tel que* `fermat(n)` *n'est pas premier. Vous afficherez aussi un diviseur premier de* `fermat(n)`*.*

□

Exercice 4 (Dessiner un tapis, ⋆⋆)

Ajoutez à votre programme les définitions de fonctions ci-dessous.

```
def star(): print('*',sep='',end='')
def sharp() : print('#',sep='',end='')
def newline() : print()

```

Sans utiliser la fonction `print`*, mais uniquement les fonctions* `star`, `sharp`, *et* `newline` *ci-dessus, écrivez des fonctions* `tapis_X(l , h)` *qui affichent des tapis de largeur* l *et de hauteur* h *avec les motifs ci-dessous.*

a)
```
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

b)
```
*#*#*#*#*#
#*#*#*#*#*
*#*#*#*#*#
#*#*#*#*#*
*#*#*#*#*#
#*#*#*#*#*
*#*#*#*#*#
#*#*#*#*#*
*#*#*#*#*#
#*#*#*#*#*
```

c)
```
**********
*########*
*########*
*########*
*########*
*########*
*########*
*########*
*########*
**********
```

d)
```
**********
*##########
**********
##########*
**********
*##########
**********
##########*
**********
*##########
```

□