Série 0 (Corrigé)

Prise en main de Jupyter et Python

Utilisez le lien `https://go.epfl.ch/analyse-num-deparis` pour vous connecter au serveur de Jupyter notebooks "noto" ou suivez les instructions sur Moodle pour utiliser Jupyter sur un ordinateur.

Ouvrez les notebooks "0.1 Jupyter tutorial", "0.2 Python tutorial" et "0.3 Some Exercises.ipynb", "0.4 Representation des nombres.ipynb"

Lisez en entier le premier pour vous familiariser avec cet outils. Vous pouvez l'éditer à votre discrétion. Ensuite passez au deuxième pour vous familiariser avec Python si nécessaire Ensuite lisez et resolvez le troisième notebook.

Ces 4 notebooks sont sourtout pour vous familiariser avec Python et Jupyter, ne passez pas plus que 45 minutes dans cette première partie. Si vous connaissez déjà ces outils, concentrez-vous sur le 4ème.

Sol. : *La solution des Jupyther notebooks se trouve sur moodle sous "analyse-numerique-chap0".*

Python

L'exercice suivant se trouve dans le notebooks *1.1 Dichotomie*. Les vidéos sont à regarder pour vendredi, ne le faites pas en salle d'exercices autrement vous dérangez les autres.

Exercice 1

Comprenez et completez la fonction suivante dans le notebook *1.1 Dichotomie* qui effectue l'algorithme de dichotomie.

Ensuite testez-la pour trouver la racine de la fonction $f(x) = x\sin(2\pi x) + \frac{1}{2}x - \frac{1}{4}$ dans l'intervalle $[-1.5, 1]$.

```
def bisection(a,b,f,tolerance,maxIterations) :
  # [a,b] interval of interest
  # f function
  # tolerance desired accuracy
  # maxIterations : maximum number of iteration
  # returns:
  # zero, residual, number of iterations
```

Sol. :

```
def bisection(a,b,fun,tol,maxIterations) :
    # [a,b] interval of interest
    # fun function
    # tolerance desired accuracy
    # maxIterations : maximum number of iteration
    # returns:
    # zero, residual, number of iterations, x_sequence
    # x_sequence : the sequence computed by the bisection

    if (a >= b) :
        print(' b must be greater than a (b > a)')
        return 0,0,0,[0]

    # what we consider as "zero"
    eps = 1e-12

    # evaluate f at the endpoints
    fa = fun(a)
    fb = fun(b)

    if abs(fa) <  eps : # a is the solution
        zero = a
        esterr = fa
        k = 0
        return zero, esterr, k, [zero]

    if abs(fb) <  eps : # b is the solution
        zero = b
        esterr = fb
        k = 0
        return zero, esterr, k, [zero]

    if fa*fb > 0 :
        print(' The sign of FUN at the extrema of the interval must be
            different')
        return 0,0,0,[0]



    # We want the final error to be smaller than tol,
    # i.e. k > log( (b-a)/tol ) / log(2) -1

    nmax = int(np.ceil(np.log( (b-a)/tol ) / np.log(2))) - 1
```

```
    # but nmax shall be smaller the the nmaximum iterations asked by the user
    if ( maxIterations < nmax ) :
        nmax = int(round(maxIterations))
        print('Warning: maxIterations is smaller than the minimum number of
            iterations necessary to reach the tolerance wished');

    # vector of intermadiate approximations etc
    x = np.zeros(nmax+1)

    # initial error is the length of the interval.
    esterr  = (b - a)

    # do not need to store all the a^k and b^k, so I call them with a new
       variable name:
    ak = a
    bk = b
    # the values of f at those points are fa and fk

    for k in range(nmax+1) :

        # approximate solution is midpoint of current interval
        x[k] =  (ak + bk) / 2
        fx = fun(x[k]);
        # error estimator is the half of the previous error
        esterr  = esterr / 2

        # if we found the solution, stop the algorithm
        if np.abs(fx) <  eps :
            # error is zero
            zero = x[k]
            esterr = 0;
            return zero, esterr, k, x

        if fx*fa < 0  :  # alpha is in (a,x)
            bk = x[k]
            fb = fx
        elif fx*fb < 0 : # alpha is in (x,b)
            ak = x[k]
            fa = fx
        else  :
            error('Algorithm not operating correctly')


    zero = x[k];

    if esterr > tol :
```

```
        print('Warning: bisection stopped without converging to the desired
            tolerance because the maximum number of iterations was reached');

    return zero, esterr, k, x
```

Vidéos

Pour ce vendredi il faut regarder et comprendre les vidéos 1.1-1.2 des notebook 1.1 et 1.2 (5 vidéos, 25 minutes, environ 50 minutes de travail).