DS3

Correction du devoir surveillé

Exercice 1

1. On peut utiliser la ligne de commande suivante :

```
import numpy as np
```

2. On peut procéder comme suit :

```
#définition de la fonction f
def f(x):
    return x**2-2

#tracé de la courbe de f
x = np.linspace(1,2,100)
y = f(x)
plt.plot(x,y)
plt.show()
```

3. La fonction `balayage` prend en entrée un entier p. Elle crée un vecteur ligne `v` de taille $10^p + 1$ dont les coefficients sont en progression arithmétique de 1 à 2. Ainsi pour $p = 3$, `v` contient le vecteur $[1, 1.001, 1.002, \dots, 1.999, 2]$. Cette fonction parcourt ensuite le vecteur `v` à l'aide d'une boucle `while`, avec pour condition d'arrêt `f(v[i])<0`. Ainsi, si la boucle `while` s'arrête au rang `i`, ce rang satisfait `f(v[i])>=0` et `f(v[i-1])<0`. Puisque f est croissante, et s'annule en changeant de signe en $\sqrt{2}$, il suit :

$$v[i-1] = v[i] - 10^{-p} < \sqrt{2} \leq v[i].$$

En renvoyant `v[i]`, la fonction `balayage` fournit une approximation de $\sqrt{2}$ à 10^{-p} près. La commande `balayage(3)` fournit donc une approximation de $\sqrt{2}$ à 10^{-3} près.

4. On peut procéder comme suit

```
def suites(n):
      a = 1
      b = 2
      for k in range(n) :
            m = (a+b)/2
            if f(a)*f(m)<=0 :
                  b = m
            else :
                  a = m
      return a, b
```

On passe n fois dans la boucle `for`. À chaque passage dans cette boucle, on calcule m, on teste si $f(a) * f(m) \leq 0$. Si c'est le cas, la variable `a` reste inchangée et `b` reçoit `m`. Sinon, c'est la variable `b` qui reste inchangée et `a` qui reçoit `m`.

5. Créons déjà deux vecteurs `a` et `b` contenant tous les termes des suites (a_n) et (b_n) pour $n = 0, \dots, 10$. Pour cela, on initialise ces vecteurs avec que des coefficients nuls. Puis, à l'aide d'une boucle `for` et de la fonction `suites`, on remplace à chaque itération le k-ème coefficient de ces vecteurs par le terme a_k et b_k.

```
a = np.zeros(11)
b = np.zeros(11)
for k in range(11):
    a[k], b[k] = suites(k)
```

Il reste alors à tracer les points de coordonnées (k, a_k) et (k, b_k) pour $k = 0, \dots, 10$, ce qu'on peut faire à l'aide de la commande `plot`. Sans oublier de créer au préalable un vecteur `x` contenant les abscisses de ces points.

```
x = np.arange(11)
plt.plot(x,a,'x')
plt.plot(x,b,'+')
plt.legend(['suite (a_n)', 'suite (b_n)']) # pour afficher une légende
plt.show()
```

Les arguments `'x'` et `'+'` permettent de ne pas relier les points et de pouvoir distinguer ceux correspondants à la suite (a_n) ou à (b_n).

À noter que la construction des vecteurs `a` et `b` comme fait plus haut nécessite à `Python` beaucoup de calculs inutiles : on calcule tous les termes des suites pour avoir le k-ème terme (c'est ce qui se passe avec l'instruction `suites(k)`), alors qu'on a déjà calculé à l'itération précédente les $(k-1)$-ème termes. Une méthode plus efficace est la suivante :

```
a = np.zeros(11) ; a[0] = 1
b = np.zeros(11) ; b[0] = 2
for k in range(1,11):
    m = (a[k-1]+b[k-1])/2
    if f(a[k-1])*f(m) <= 0 :
        b[k] = m
    else :
        a[k] = m
```

6. Puisque $|\sqrt{2}-a_n| \leq |b_n-a_n|$, il suffit de trouver un rang n pour lequel $|b_n-a_n| \leq \varepsilon$, et de renvoyer a_n qui fournira alors une approximation de $\sqrt{2}$ à ε près. On va pour cela reprendre en partie le programme précédent. On initialise les variables `a` à $a_0=1$ et `b` à $b_0=2$. Et tant que $|b_n-a_n|>\varepsilon$, on passe au rang suivant. On va donc effectuer une boucle `while`. Puisque les suites (a_n) et (b_n) sont adjacentes, on a la certitude que cette boucle s'arrêtera bien à un certain rang.

   Voici une possibilité de script.

```
def dichotomie(eps):
      a = 1
      b = 2
      while np.abs(b-a) >= eps :
            m = (a+b)/2
            if f(a)*f(m)<=0 :
                  b = m
            else :
                  a = m
      return a
```

7. On garde la même trame en ajoutant une variable `n` qui comptera le nombre d'itérations nécessaires, en prenant pour la condition d'arrêt `eps = 10**(-p)`

```
def etapes(p):
      a = 1
      b = 2
      n = 0
      while np.abs(b-a) >= 10**(-p) :
            m = (a+b)/2
            n = n+1
            if f(a)*f(m)<=0 :
                  b = m
            else :
                  a = m
      return n
```

8. On peut procéder ainsi :

```
A = np.ones((10,10)) + np.eye(10,10)
```

9. On peut par exemple demander le rang de A et s'assurer qu'il est bien égal à 10 :

```
print(al.matrix_rank(A)==10)
```

10. On peut procéder comme suit :

```
def suite(n):
    Y = np.ones((10,10))+np.eye(10,10)
    Z = np.eye(10,10)
    for k in range(n):
        M = (Y+al.inv(Z))/2
        Z = (Z+al.inv(Y))/2
        Y = M
    return Y
```

11. Afin de vérifier la pertinence du résultat renvoyé par la fonction `suite`, on peut utiliser la commande suivante :

```
print(al.matrix_power(suite(2000),2))
```

On pouvait également procéder ainsi :

```
print(np.dot(suite(2000), suite(2000)))
```

Rappels sur la méthode de dichotomie

Considérons une fonction $f : [a,b] \to \mathbb{R}$ **continue** et supposons que f **s'annule en une seule et unique valeur** $\alpha \in]a,b[$ et **y change de signe**. On ne peut pas forcément calculer explicitement α et on doit parfois utiliser des méthodes de calcul approché pour avoir une estimation de la valeur de α. Une méthode bien connue est le principe de dichotomie. Comme f s'annule et change de signe, elle satisfait en particulier $f(a)f(b) < 0$.

Principe. On considère $m = \dfrac{a+b}{2}$ le milieu du segment $[a,b]$. Alors deux cas sont possibles :

- soit $f(a)f(m) \leq 0$: alors $f(a)$ et $f(m)$ sont de signe opposé. Comme f est continue, f doit s'annuler sur $[a,m]$ et donc $\alpha \in [a,m]$.
- soit $f(m)f(b) \leq 0$: alors $f(m)$ et $f(b)$ sont de signe opposé et de la même façon, $\alpha \in [m,b]$.

On recommence alors le raisonnement précédent avec l'intervalle obtenu contenant α.

On construit une suite d'intervalles $[a_n,b_n]$ contenant α : Pour cela, on définit trois suites $(a_n)_{n\in\mathbb{N}}$, $(b_n)_{n\in\mathbb{N}}$, $(m_n)_{n\in\mathbb{N}^*}$ par $a_0 = a$, $b_0 = b$ et pour tout $n \in \mathbb{N}$, $m_{n+1} = \dfrac{a_n+b_n}{2}$ et

- si $f(a_n)f(m_{n+1}) \leq 0$, alors $\begin{cases} a_{n+1} = a_n \\ b_{n+1} = m_{n+1} \end{cases}$
- si $f(m_{n+1})f(b_n) \leq 0$, alors $\begin{cases} a_{n+1} = m_{n+1} \\ b_{n+1} = b_n \end{cases}$

Propriété (Méthode de la dichotomie)

Les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ sont adjacentes et elles convergent vers l'unique zéro α de f sur $]a,b[$.

Preuve. Commençons pour cela par montrer par récurrence que pour tout $n \in \mathbb{N}$:

$$a_n \leq a_{n+1} \leq b_{n+1} \leq b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)f(b_n) \leq 0.$$

Init. Par construction, on a soit $a_1 = a_0$ et $b_1 = \dfrac{a_0+b_0}{2}$, soit $a_1 = \dfrac{a_0+b_0}{2}$ et $b_1 = b_0$. Comme $a_0 = a < b = b_0$, on a ainsi l'inégalité $a_0 \leq a_1 \leq b_1 \leq b_0$. D'autre part, $b_0 - a_0 = b - a = \dfrac{b-a}{2^0}$ et $f(a_0)f(b_0) = f(a)f(b) \leq 0$ par hypothèse. Donc la propriété est vraie au rang $n = 0$.

Hér. Soit n un entier naturel. Supposons la propriété au rang n vraie. Montrons la propriété au rang $n+1$.

Par construction, on a soit $a_{n+2} = a_{n+1}$ et $b_{n+2} = \dfrac{a_{n+1}+b_{n+1}}{2}$, soit $a_{n+2} = \dfrac{a_{n+1}+b_{n+1}}{2}$. Par hypothèse de récurrence, $a_{n+1} \leq b_{n+1}$. On a donc dans les deux cas l'inégalité $a_{n+1} \leq a_{n+2} \leq b_{n+2} \leq b_{n+1}$.

De plus, par construction, on a $b_{n+1} - a_{n+1} = \dfrac{b_n - a_n}{2}$. Par hypothèse de récurrence, $b_n - a_n = \dfrac{b-a}{2^n}$. Donc $b_{n+1} - a_{n+1} = \dfrac{b-a}{2^{n+1}}$.

Enfin, comme par hypothèse de récurrence $f(a_n)f(b_n) \leq 0$, $f(a_n)$ et $f(b_n)$ sont de signe opposé. Donc $f(m_{n+1}) = f(\dfrac{a_n+b_n}{2})$ n'a pas le même signe que $f(a_n)$ ou $f(b_n)$. Par définition de a_{n+1} et de b_{n+1}, on a bien $f(a_{n+1})f(b_{n+1}) \leq 0$.

Donc la propriété au rang $n+1$ est vraie.

Par le principe de récurrence, on a bien pour tout $n \in \mathbb{N}$:

$$a_n \leq a_{n+1} \leq b_{n+1} \leq b_n, \quad b_n - a_n = \frac{b-a}{2^n} \quad \text{et} \quad f(a_n)f(b_n) \leq 0.$$

Ainsi, (a_n) est croissante, (b_n) est décroissante, et : $b_n - a_n = \dfrac{b-a}{2^n} \underset{n\to+\infty}{\longrightarrow} 0$.
Les suites (a_n) et (b_n) sont donc adjacentes et, d'après le théorème des suites adjacentes, convergent vers la même limite $\ell \in]a, b[$.

Reste enfin à montrer que $\ell = \alpha$. En passant à la limite dans l'inégalité $f(a_n)f(b_n) \leq 0$, on obtient (en utilisant que f est continue sur $[a, b]$) :

$$\lim_{n\to+\infty} (f(a_n) \cdot f(b_n)) = \left(\lim_{n\to+\infty} f(a_n)\right)\left(\lim_{n\to+\infty} f(b_n)\right) = (f(\ell))^2 \leq 0.$$

D'où $(f(\ell))^2 \leq 0$, ce qui implique $f(\ell) = 0$. Et comme par hypothèse f s'annule en une seule valeur $\alpha \in]a, b[$, on a donc $\ell = \alpha$. □