KORREKTUR. TD PYTHON 12

Autour des polynômes de Lagrange

Exercice 01 : manipulation de polynômes

Soit le polynôme $P = X^3 + 2X - 3$, calculer successivement $P(0)$, $[P(1), P(2), P(3)]$, deg P, les racines de P, puis les polynômes P', P'' et enfin le quotient et le reste de la division euclidienne de P par $B = X^2 + 1$.

```
from numpy.polynomial import Polynomial
P = Polynomial([-3,2,0,1])
P(0), P([1,2,3]), P.degree()
     -3.0 , array([0., 9., 30.])  ,  3
P.roots(), P.deriv().coef, P.deriv(2).coef
    array([-0.5-1.6583124j, -0.5+1.6583124j, 1.0 + 0.j ]) ,
    array([2., 0., 3. ]) , array([0., 6.])
B = Polynomial([1, 0, 1 ]; Q = P // B ; R = P % B
Q.coef, R.coef
     array([0., 1.]) , array([-3., 1.])
```

Exercice 02 : Un premier exemple

Déterminer la base d'interpolation de Lagrange pour la famille $(0, 1, 2)$.
En déduire l'unique polynôme P de degré 2 tel que $P(0) = 1$, $P(1) = -3$ et $P(2) = 2$.

Solution

Appelons L_0, L_1 et L_2 les trois polynômes de la base d'interpolation associée à $[0, 1, 2]$. On a :

$$L_0 = \frac{(X-1)(X-2)}{(0-1)(0-2)},\ L_1 = \frac{(X-0)(X-2)}{(1-0)(1-2)},\ L_2 = \frac{(X-0)(X-1)}{(2-0)(2-1)}.$$

Ce qui donne :

$$L_0 = 1 - \frac{3}{2}X + \frac{1}{2}X^2, L_1 = 2X - X^2, L_2 = -\frac{1}{2}X + \frac{1}{2}X^2.$$

Et en fin, $P = 1 \times L_0 - 3 \times L_1 + 2 \times L_2$, c'est-à-dire :

$$P = 1 - \frac{3}{2}X + \frac{1}{2}X^2 - 3\left(2X - X^2\right) - X + X^2 = 1 - \frac{17}{2}X + \frac{9}{2}X^2.$$

Exercice 03

1. Créer une fonction `BASE` d'argument la liste $Xabs$, qui renvoie la liste des polynômes de la base d'interpolation de Lagrange associée aux valeurs de la liste $Xabs$. Appliquer au cas où $Xabs = [0, 1, 2]$ et retrouver le résultat de l'exercice 02.

2. En utilisant `BASE` comme sous-procédure, créer une fonction `POL_LAG` d'arguments $Xabs$ et f qui renvoie le polynôme d'interpolation de Lagrange passant par les points dont les abscisses sont les valeurs de X et les ordonnées sont les images par f des valeurs de $Xabs$.
Appliquer avec $Xabs = [0, 1, 2]$ et la fonction $x \mapsto x^2$. Que remarque t-on ?
Appliquer avec $Xabs = [-1, 0, 1, 2, 3]$ et $f = \arctan$. On appellera dans la suite P_1 le dernier polynôme trouvé.

3. Tracer ensemble P_1 et la fonction arctan sur $[-1, 3]$. On pourra différencier les graphes avec l'option `linestyle`.

4. Retrouver P_1 en utilisant `interpolate.lagrange` de `scipy`.

Solution

1. On tape :

```
from numpy.polynomial import Polynomial
def BASE(Xabs):
   n = len(X)-1; L=[]
   for j in range(0,n+1):
      P = Polynomial([1])
      for i in range(0,n+1):
         if i!= j :
            P = P*Polynomial([-Xabs[i]/(Xabs[j]-Xabs[i]),
            1/(Xabs[j]-Xabs[i])])
      L.append(P)
   return L
```

On applique donc avec $Xabs = [0, 1, 2]$ et on obtient 3 polynômes du second degré.

```
BASE([0,1,2])
[Polynomial([ 1. , -1.5,  0.5],
 domain=[-1.,  1.], window=[-1.,  1.]),
 Polynomial([ 0.,  2., -1.],
 domain=[-1.,  1.], window=[-1.,  1.]),
 Polynomial([ 0. , -0.5,  0.5],
 domain=[-1.,  1.], window=[-1.,  1.])]
```

On obtient bien le même résultat que dans l'exercice 02.

2.

```
def POL_LAG(Xabs,f):
   n = len(Xabs) - 1
   return sum(f(Xabs[i])*BASE(Xabs)[i] for i in range(0,n+1))
```

On applique pour $X = [0, 1, 2]$ et $f(x) = x^2$:

```
def f(x) : return x**2
POL_LAG([0,1,2],f)

  Polynomial([0., 0., 1.], domain=[-1.,  1.], window=[-1.,  1.])
```

On trouve que le polynôme d'interpolation de degré 2 est la fonction f, ce qui est normal car f est aussi une fonction polynomiale de degré 2.
On cherche now le polynôme d'interpolation passant par $[-1, 0, 1, 2, 3]$ et l'image par arctan.

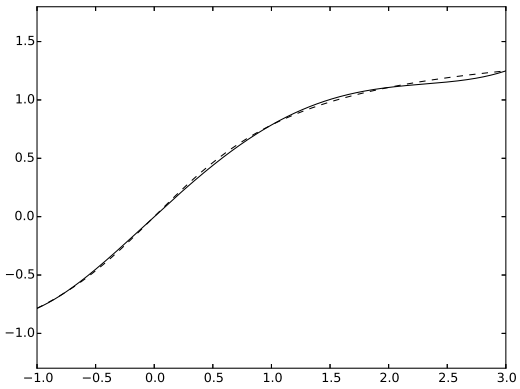
```
POL_LAG([-1,0,1,2,3],np.arctan)
       Polynomial([ 0.        ,  0.92495957, -0.0311434 , -0.13956141,
          0.0311434 ], domain=[-1.,  1.], window=[-1.,  1.])
```

Juste pour vérifier ! :

```
POL_LAG([-1,0,1,2,3],np.arctan)(2)
       1.1071487177940906
np.arctan(2)
       1.1071487177940904
```

3. Ici, on note XX l'intervalle $[-1,3]$ pour ne pas le confondre avec $Xabs$ qui n'a que 5 valeurs. Ne pas oublier de taper les `plt`

```
P1 = POL_LAG([-1,0,1,2,3],np.arctan)
import matplotlib.pyplot as plt
XX = np.linspace(-1,3,500)
plt.plot(XX,np.arctan(XX),linestyle="- -");
plt.plot(XX,P1(XX),color='0');
plt.axis('equal'); plt.show()
```



4. Warnung, on obtient $P1$ par puissances décroissantes.

```
from scipy import interpolate
X = np.array([-1,0,1,2,3]); Y=np.arctan(X)
import scipy as sp
P1 = sp.interpolate.lagrange(X,Y)
P1
  poly1d([ 0.0311434 , -0.13956141, -0.0311434 ,  0.92495957,  0.])
```

Exercice 04

Soit $n \in \mathbf{N}^\star$, on fixe $n+1$ points distincts $a \leqslant x_0 < x_1 < ... < x_{n-1} < x_n \leqslant b$.
Pour tout $j \in [\![0,n]\!]$, on note L_j le $j^{\text{ème}}$ ***polynôme de la base d'interpolation de Lagrange*** associé à $(x_0,...,x_n)$. En fixant encore $n+1$ points quelconques $y_0,...,y_n$, on rappelle que le polynôme $L = \sum_{j=0}^{n} y_j L_j$ est le ***polynôme de Lagrange*** associé à $((x_0,y_0),...,(x_n,y_n))$.

1. Créer en Python une fonction `Lagrange` qui a pour argument trois listes $X = [x_0,...,x_n]$, $Y = [y_0,...,y_n]$ et $T = [t_0,...,t_p]$, où les x_i sont deux à deux distincts et qui renvoie la liste $[L(t_0),...L(t_p)]$, où L est le polynôme de Lagrange plus haut.

2. Ici on suppose que $a = 0$, $b = 2$ et pour tout $j \in [\![0,n]\!]$, $x_j = j\dfrac{2}{n}$.
Tracer ensemble sur $[0,2]$, le graphe de $f : x \mapsto e^{-x^2}$ et les polynômes de Lagrange associés au support $((x_0,f(x_0)),...,(x_n,f(x_n))$ pour tous les entiers n appartenant à $[\![1,19]\!]$.

Solution

1. Ici, dans la procédure, on n'a pas besoin de construire le polynôme L en tout X. Ici $n = len(X)$ correspond à $n+1$ de l'énoncé. Puis `Liste[j]` correspond à `L_j(t_k)`.

```
def Lagrange(X,Y,T) :
    n = len(X)-1; s = 0; Nliste=[ ]; p = len(T)
    for k in range(0,p):
        Liste = [ ]
        for j in range(0,n+1) :
            q = 1
            for i in range(0, n+1):
                if i != j:
                   q=q*(T[k]-X[i])/(X[j]-X[i])
            Liste.append(q)
        s = sum(Y[l]*Liste[l] for l in range(0,n+1))
        Nliste.append(s)
     return(Nliste)
```

2. Ici, on commence donc par créer une fonction polynomiale de Lagrange que l'on « plotera » plus loin.

```
def Fonct_Lagrange(X,Y,x) :
    n = len(X)-1; s = 0; L=[]
    for j in range(0,n+1) :
        q = 1
        for i in range(0,n+1) :
            if i!= j :
               q = q*(x-X[i])/(X[j]-X[i])
        L.append(q)
    return sum(Y[l]*L[l] for l in range(0,n+1))
```

Il reste à tracer tout cela. Ici XX est la liste des points pour le tracé de toutes les courbes et X est la liste $[x_0, ..., x_n]$ qui permet de construire le polynôme de Lagrange.

```
import matplotlib.pyplot as plt
import numpy as np
def f(x) : return np.exp(-x**2)
XX = np.linspace(0,2,200)
for n in range(1,20) :
    X = np.linspace(0,2,n+1)
    Y = f(X); plt.plot(XX,f(XX));
    plt.plot(XX,Fonct_Lagrange(X,Y,XX)); plt.show()
```