

TP 4 - éléments de correction

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.interpolate as inter
4
5 ## Exemple 1
6 x = np.array([0,1,2])
7 w = np.array([4,-2,4.])
8 p = inter.lagrange(x,w)
9 print(p)
10 print(p(0), p(1), p(2))
11 xpt=np.linspace(-1,3,40)
12 plt.plot(x,w,'+',label = "points d'interpolation")
13 plt.plot(xpt,p(xpt),'k',label = "polynôme d'interpolation (lagrange)")
14 plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.', label = "polynôme d'interpolation (barycentric_interpolate)")
15 plt.legend()
16 plt.show()
17
18 ## Exemple 2
19 x=np.array([-4,0,1,4])
20 w=np.cos(x)
21 p=inter.lagrange(x,w)
22 print(p)
23 print(p(-4), p(0), p(1), p(4))
24 xpt=np.linspace(-4,4,40)
25 plt.clf()
26 plt.plot(x,w,'+',label="points d'interpolation")
27 plt.plot(xpt,np.cos(xpt),'k',label="fonction à interpoler (cosinus)")
28 plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.',label="polynôme d'interpolation (barycentric_interpolate)")
29 plt.legend()
30 plt.show()
31
32 ## 1.1. Premier test
33 x = np.array([0,1,2,3,4])
34 w = np.sin(x)
35 p = inter.lagrange(x,w)
36 print(f"valeurs aux points d'interpolation avec routine lagrange :\n{p(x)}")
37 plt.clf()
38 plt.plot(x,w,'bo',label="points d'interpolation")
39 xpt = np.linspace(-2,4,40)
40 plt.plot(xpt,np.sin(xpt),'go',label = "fonction à interpoler (sinus)")
41 plt.plot(xpt,p(xpt),'k',label="polynôme d'interpolation (lagrange)")
42 print(f"valeurs aux points d'interpolation avec barycentric_interpolate :\n {inter.barycentric_interpolate(x,w,x)}")
43 plt.plot(xpt,inter.barycentric_interpolate(x,w,xpt),'r-.',label="polynôme d'interpolation (barycentric_interpolate)")
44 plt.legend()
45 plt.show()
46
47 ## 1.2. Exercice
48 def f(x):
49     return x**(1/3)
50
51 # a)
52 x = np.array([0,1,8,27,64])
53 w = f(x)
54 p = inter.lagrange(x,w)
55 print(f"a) Le polynôme d'interpolation de Lagrange est, selon la routine lagrange : \nP(x) = {p}")
56
57 # b)
58 print(f"b) P(20) = {p(20)}\n20**(1/3) = {20**(1/3)}")
59
60 # c)
61 xpt = np.linspace(0,64,200)
62 plt.clf()
63 plt.plot(x,w,'o',label="points d'interpolation")
64 plt.plot(xpt,f(xpt),'g',label="fonction à interpoler (racine cubique)")
65 plt.plot(xpt,p(xpt),'r-.',label = "polynôme d'interpolation (lagrange)")
66
67 # d)
68 x_sans0 = np.array([1,8,27,64])
69 w_sans0 = f(x_sans0)
70 p_sans0 = inter.lagrange(x_sans0,w_sans0)
```

```

71 print(w_sans0)
72 print(p_sans0(x_sans0))
73 plt.plot(xpt,p_sans0(xpt),'k',label = "polynôme d'interpolation sans (0,0) (lagrange)")
74 plt.legend()
75 plt.show()
76
77 ## 2.1. Points équidistants
78 def einterp(n,f,a,b):
79     plt.clf()
80     xpt = np.linspace(a,b,1000)
81     plt.plot(xpt,f(xpt),'g',label="f")
82     x = np.linspace(a,b,n+1)
83     plt.plot(xpt,inter.barycentric_interpolate(x,f(x),xpt),'k-.',label = "f interpolée")
84     plt.plot(x,f(x),'x',label="points d'interpolation")
85     plt.legend()
86     plt.show()
87
88 a , b = -1 , 1
89 def f1(x):
90     return 1/(1+x**2)
91 n = 57
92 einterp(n,f1,a,b)
93
94 def f2(x):
95     return 1/(4/25 +x**2)
96 n = 20
97 einterp(n,f2,a,b)
98
99 ## 2.2. Points de Tchebychev
100 def tinterp(n,f,a,b):
101     plt.clf()
102     xpt = np.linspace(a,b,1000)
103     plt.plot(xpt,f(xpt),'g',label="f")
104     x = np.zeros(n+1)
105     for k in range(0,n+1):
106         x[k] = (a+b)/2 + (b-a)/2 * np.cos( np.pi * (2 * k + 1)/(2*(n+1)))
107     plt.plot(xpt,inter.barycentric_interpolate(x,f(x),xpt),'k-.',label = "f interpolée")
108     plt.plot(x,f(x),'x',label="points d'interpolation")
109     plt.legend()
110     plt.show()
111
112 a , b = -1 , 1
113 def f1(x):
114     return 1/(1+x**2)
115 n = 60
116 tinterp(n,f1,a,b)
117
118 def f2(x):
119     return 1/(4/25 +x**2)
120 n = 20
121 tinterp(n,f2,a,b)

```