Rendue : Moca S8-S9 Lazrak Ali 28605235

Exercice 1 :

import numpy as np

import matplotlib.pyplot as plt

def f(x):

    return 1/(1+25\*pow(x, 2))

x = np.linspace(-1, 1, 500)

y = f(x)

plt.plot(x, y, color="blue")

x2 = []

y2 = []

for i in range(10):

    x2.append(-1+i/5)

    y2.append(f(-1+i/5))

x2 = np.array(x2)

y2 = np.array(y2)

plt.scatter(x2, y2)

xp = []

yp = []

for i in range(10):

    xp.append(-1+i/5)

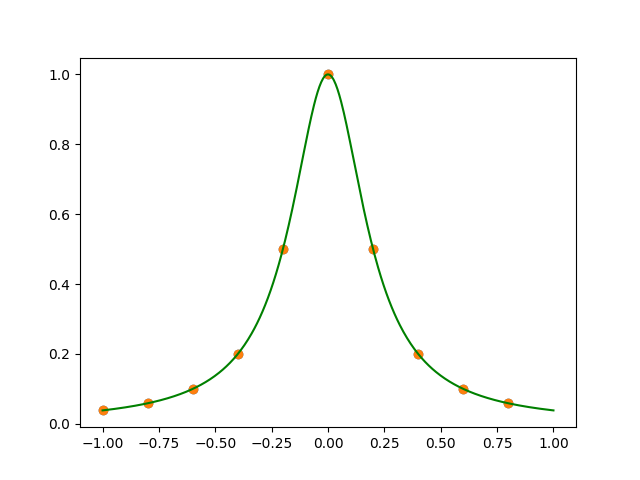
    yp.append(f(-1+i/5))

xp = np.array(xp)

yp = np.array(yp)

plt.scatter(xp, yp)

plt.show()



def l(x, xp, i):

    res=1

    for xj in xp:

        if xp[i] != xj:

            res = res \* ((x-xj)/(xp[i] - xj))

    return res

def lagrange(x, xp, yp):

    res = 0

    for i in range(len(xp)):

        res += yp[i] \* l(x, xp, i)

    return res

x = np.linspace(-1,1,500)

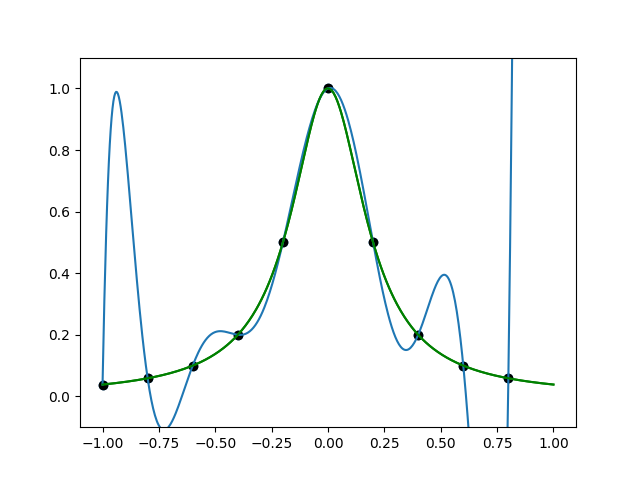
y = lagrange(x, xp, yp)

plt.ylim((-0.1, 1.1))

plt.plot(x, y)

plt.scatter(xp, yp, color="black")

plt.plot(x, f(x), "green")



from pulp import \*

def s(x, xi, ai, bi, ci, di):

    return ai + bi \* (x - xi) + ci \* (x - xi)\*\*2 + di \* (x - xi)\*\*3

def s\_p(x, xi, bi, ci, di):

    return bi + ci \* 2 \* (x-xi) + di \* 3 \* (x - xi) \*\* 2

def s\_pp(x, xi, ci, di):

    return 2 \* ci + 6 \* di \* (x-xi)

model = LpProblem(name="Spline Cubique")

a = LpVariable.matrix("a", list(range(1,len(xp)+1)))

b = LpVariable.matrix("b", list(range(1,len(xp)+1)))

c = LpVariable.matrix("c", list(range(1,len(xp)+1)))

d = LpVariable.matrix("d", list(range(1,len(xp)+1)))

model += 1

#x = np.linspace(-1,1,10)

for i in range(len(xp) - 1):

    model += s(xp[i], xp[i],a[i],b[i],c[i],d[i]) == yp[i]

    model += s(xp[i+1], xp[i],a[i],b[i],c[i],d[i]) == yp[i+1]

for i in range(len(xp) - 2):

    model += s\_p(xp[i+1], xp[i],b[i],c[i],d[i]) == s\_p(xp[i+1], xp[i+1],b[i+1],c[i+1],d[i+1])

for i in range(len(xp) - 2):

    model += s\_pp(xp[i+1], xp[i],c[i],d[i]) == s\_pp(xp[i+1], xp[i+1],c[i+1],d[i+1])

model += s\_p(xp[0], xp[0], b[0], c[0], d[0]) == -(50 \* x[0])/((1 + 25\*x[0])\*\*2)

model += s\_p(xp[8], xp[9], b[8], c[8], d[8]) == -(50 \* x[9])/((1 + 25\*x[9])\*\*2)

# Résolution du problème

# keepFiles permet de créer un fichier .sol contenant la solution

status = model.solve(solver=GLPK(msg=True, keepFiles=False))

# Affiche le statut de la solution (optimale, non borné, etc.)

print("Status:", LpStatus[model.status])

# Affiche la valeur de la fonction objectif

print("objective=", value(model.objective))

# Affiche les valeurs optimales des variables de décision :

for i in range(len(xp) - 1):

    print("a =", value(a[i]))

    print("b =", value(b[i]))

    print("c =", value(c[i]))

    print("d =", value(d[i]))

values = xp

for i in range(len(values) - 1):

    x\_axis = np.linspace(values[i], values[i+1], 200)

    plt.plot(x\_axis, s(x\_axis, values[i], value(a[i]), value(b[i]),value(c[i]),value(d[i])))

affichage :

GLPSOL: GLPK LP/MIP Solver, v4.65

Parameter(s) specified in the command line:

 --cpxlp C:\Users\lazra\AppData\Local\Temp\962955d9079b4973ae5f0fba02da481f-pulp.lp

 -o C:\Users\lazra\AppData\Local\Temp\962955d9079b4973ae5f0fba02da481f-pulp.sol

Reading problem data from 'C:\Users\lazra\AppData\Local\Temp\962955d9079b4973ae5f0fba02da481f-pulp.lp'...

36 rows, 37 columns, 105 non-zeros

79 lines were read

GLPK Simplex Optimizer, v4.65

36 rows, 37 columns, 105 non-zeros

Preprocessing...

26 rows, 26 columns, 84 non-zeros

Scaling...

 A: min|aij| =  8.000e-03  max|aij| =  2.000e+00  ratio =  2.500e+02

GM: min|aij| =  7.383e-01  max|aij| =  1.354e+00  ratio =  1.835e+00

EQ: min|aij| =  5.473e-01  max|aij| =  1.000e+00  ratio =  1.827e+00

Constructing initial basis...

Size of triangular part is 21

      0: obj =   0.000000000e+00 inf =   1.827e+03 (5)

      5: obj =   0.000000000e+00 inf =   0.000e+00 (0)

OPTIMAL LP SOLUTION FOUND

Time used:   0.0 secs

Memory used: 0.1 Mb (66733 bytes)

Writing basic solution to 'C:\Users\lazra\AppData\Local\Temp\962955d9079b4973ae5f0fba02da481f-pulp.sol'...

Status: Optimal

objective= None

a = 0.0384615

b = 0.0868056

.

.

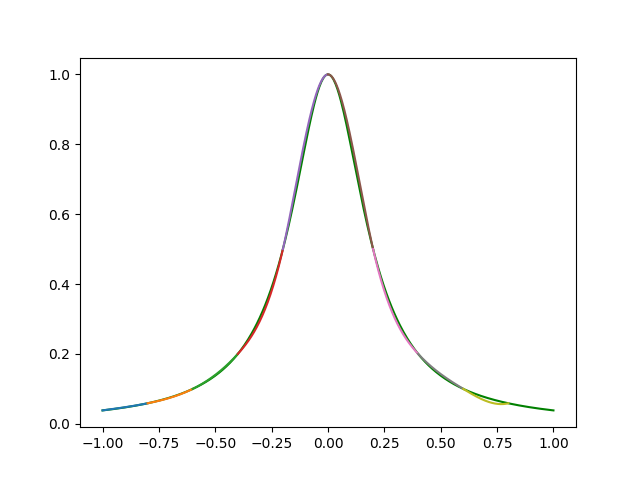
.

a = 0.1

b = -0.394544

c = 0.0811059

d = 4.31102



Exercice 2 :

Q.1)

def B(x,i,k,u):

    if k == 0:

        if u[i]<=x<u[i+1]:

            return 1.0

        else:

            0.0

    if u[i+k]==u[i]:

        c1=0.0

    else:

        c1=(x-u[i])/(u[i+k]-u[i])\*B(x,i,k-1,u)

    if u[i+k+1] == u[i+1]:

        c2=0.0

    else:

        c2=(u[i+k+1]-x)/(u[i+k+1]-u[i+1])\*B(x,i+1,k-1,u)

    return c1+c2

def bcurve(x,u,k,c):

    assert(len(c) <= len(u))

    n = len(c)

    somme = 0

    for i in range(n):

        somme += c[i]\*B(x,i,k,u)

    return somme

x=np.linspace(0,5,100)

u=[0,0,0,0,1,2,3,4,5,5,5,5]

k=3

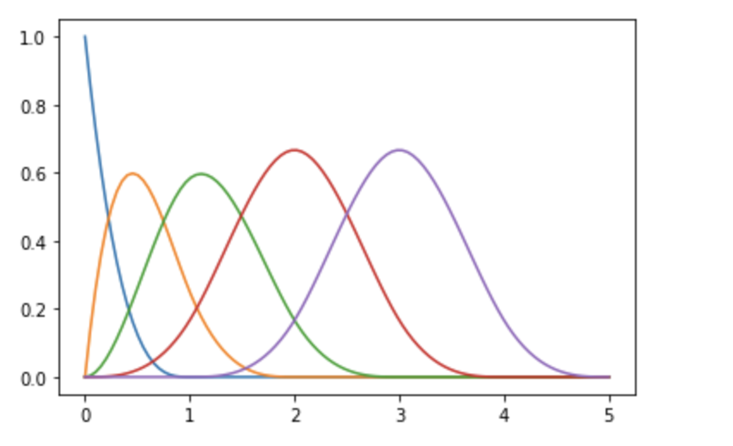
for i in range(5):

    y= [B(x,i,k,u) for x in np.linspace(0,5,100)]

    plt.plot(x,y)

plt.show()

affichage :



    Question 3:

    # Evalue la I-spline d'ordre k en x

    # direcuemenu en foncuion des Bspline pluuou que comme l'inu de la m-spline de degré k)

def I(x,i,k,u):

    for j in range(len(u)):

        if u[j] <= x <= u[j+1]:

            break

    if j < i+1:

        return 0

    if j > i+k:

        return 1

    s=0

    for t in range(i+1,j+1):

        s += B(x,t,k,u)

    return s

def icurve(x,u,k,c):

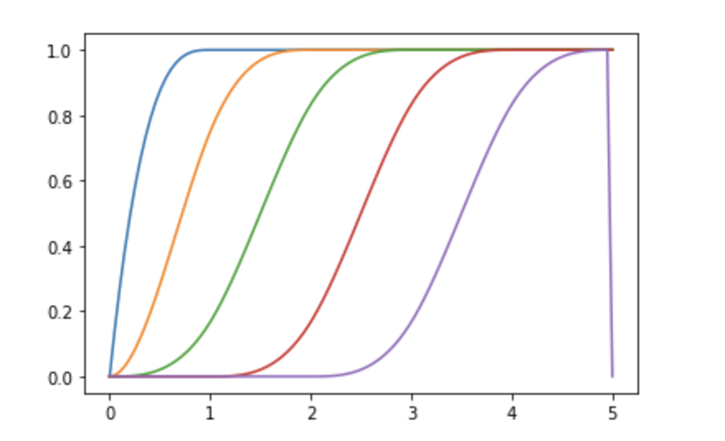
    s = 0

    for i in range(len(c)):

        s += c[i] \* I(x,i,k,u)

    return s

affichage :



p#Question 3:

# Evalue la I-spline d'ordre k en x

# direcuemenu en foncuion des Bspline pluuou que comme l'inu de la m-spline de degré k)

def I(x,i,k,u):

    for j in range(len(u)):

        if u[j] <= x <= u[j+1]:

            break

    if j < i+1:

        return 0

    if j > i+k:

        return 1

    s=0

    for t in range(i+1,j+1):

        s += B(x,t,k,u)

    return s

def icurve(x,u,k,c):

    s = 0

    for i in range(len(c)):

        s += c[i] \* I(x,i,k,u)

    return s

x=np.linspace(0,5,100)

u=[0,0,0,0,1,2,3,4,5,5,5,5]

c=[0,0,0.3,0.3,0.4]

k=3

for i in range(5):

    y= [I(x,i,k,u) for x in np.linspace(0,5,100)]

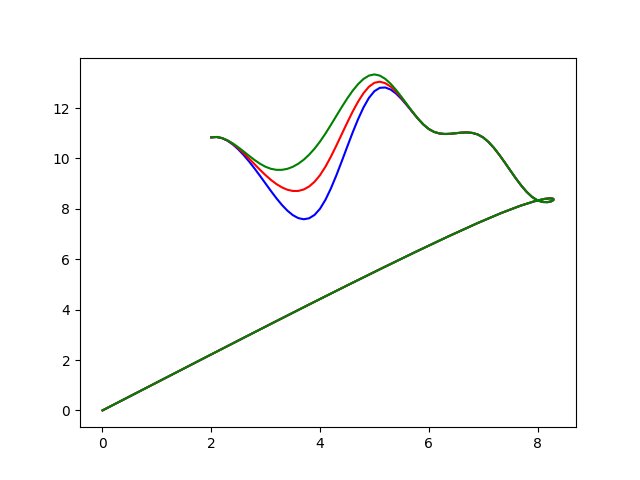
    plt.plot(x,y)

plt.show()

plt.plot(xspline,ysplne,'r')#affichage nouvelle courbe (4,6)

plt.plot(xspline1,ysplne1,'g')#affichage nouvelle courbe(4,8)

plt.show()



Exercice 4:

xi=[18,5,4,6,19,10,19,12,20,9,9,2,14,7,15,4,8,16,4,1]

yi=[14,9,8,8,17,15,19,14,18,15,14,10,7,10,11,8,13,8,8,10]

k= 3

#On calcule alpha\_i

    #On genere des fichiers .lp qui on pour objectif de minimiser la somme des uj+vj. Avec pour contrainte l'egalite

yj = uj - vj + somme{i,m-k}(alpha\_i \* Bspline{i,k}(xj))

#On resoud le programme lineaire pour avoir le coef alpha\_i

#On genere le fichier .lp avec

f = open('pas5.lp','w')

f.write("Minimize\n")

s = "\t"

for i in range(20):

    s+='u{1} + v{1} + '.format(i,i)

f.write(s)

f.write("\nSubject To")

s = ""

for i in range(20):

    s='\tc{0}: u{1} - v{1} + '.format(i+1,i)

    B\_xi = [ (B(xi[i],j,k,u), "a"+str(j)) for j in range(8)]

    for (x,a) in B\_xi:

        if (x!=0):

            s+='{0} {1} + '.format(x,a)

    s=s[0:-2]

    s+='= {0}'.format(yi[i])

    f.write(s + '\n')

f.close()

#Une fois les m-k alpha\_i on peut tracer la courbe a l'aide de bcurve;

#Noued de pas 5:

    #Ce qui differe entre les 2 regressions sont les aplha\_i et les noeuds (car on chande de pas)

u= range(-10,41,5)

alpha= [16.3545, 2.2875, 22.6696, -2.96578, 28.8215, -4.32022, 0]

xPrint=[i for i in np.arange(0,max(xi),0.1)]

yPrint=[bcurve(i,u,k,alpha) for i in np.arange(0,max(xi),0.1)]

print(distanceMoyenne(listTuplePts(xi,yi), listTuplePts(xPrint,yPrint)))#calcule et print distance moyenne de points et de la courbe

plt.plot(xi,yi,'db')

plt.plot(xPrint,yPrint,'r')

plt.show()

#on passe de 7 a 8 alpha\_i lorsque change le pas a 3 nbAlpha\_i =m-k

#Noeud de pas 3

u=range(-6,31,3)

alpha=[ 11.3748, 9.27877, 5.4163, 17.0977, 16.193, 2.13048,15.7862, 21.1473]

xPrint=[i for i in np.arange(0,max(xi),0.1)]

yPrint=[bcurve(i,u,k,alpha) for i in np.arange(0,max(xi),0.1)]

print(distanceMoyenne(listTuplePts(xi,yi), listTuplePts(xPrint,yPrint))) #calcule et print distance moyenne de points et de la courbe

plt.plot(xi,yi,'db')

plt.plot(xPrint, yPrint,'r')

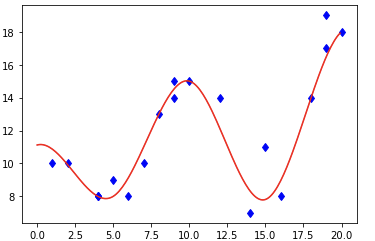
plt.show()

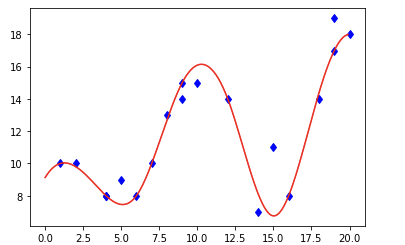
#on trouve comme distance moyenne entre les points et la courbe:

    #pas 5: 8.92e-06

    #pas 3: 0.01296108232973452

#affichage pour pas 5 :

#affichage pour pas 3 :



    Exercice 5:

    Question 1:

    #Noeud de pas 4

    xi=[1,6,8,12,10,14,15,17,20,23,3,4,7,12,13,16,18,19,20,21]

    yi=[14,15,14,11,14,6,7,8,6,2,12,14,13,5,7,8,7,6,5,3]

    k=3

    u= range(-15,41,4)

    print(len(xi))

    alpha=[0,44.3326,5.71491,16.8078,13.645,4.8132,9.10217,3.46308]

    xspline=[i for i in np.arange(0,max(xi),0.1)]

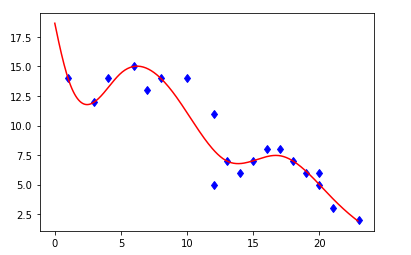
    yspline=[bcurve(i,u,k,alpha) for i in np.arange(0,max(xi),0.1)]

    plt.plot(xi,yi,'db')

    plt.plot(xspline,yspline,'r')

    plt.show()

    Question 2:



    Exercice 6:

        #Tracage de la spirale

    k=3 # Spline cubique

    x\_points\_figure=[18,11,4,10,16,10,7,11,14,12,10] # on prend la liste des X de la spirale

    y\_points\_figure=[4,2,8,16,8,6,10,14,10,8,10] # on prend la liste des Y de la spirale

    x\_points\_figure1=[18,18,11,4,10,16,10,7,11,14,12,10,10] #on ajoute les k points

    y\_points\_figure1=[4,4,2,8,16,8,6,10,14,10,8,10,10] #on ajoute les k points

    u=[j for j in range(-k,len(x\_points\_figure)+k+1)]# on crée les noeuds

    C = np.array([[0] \* 13 for x in range(13)]) # on crée la matrice B de taille m+k

    for i in range(len(C)): #on initialise la matrice B

        for j in range(len(C[i])):

            if i==j:

                C[i][j]=4

            elif i-j<2 and i-j>-2:

                C[i][j] =1

    C=C.dot(0.166666667) # On multiplie la matrice B par 1/6

    B\_inv=np.linalg.inv(C) # on l'inverse

    Q=[[x\_points\_figure1[i],y\_points\_figure1[i]] for i in range(len(x\_points\_figure1))] # crée la matrice des points X et Y de la spirale

    P=B\_inv.dot(Q) #on calcule P

    Px=[P[x][0] for x in range(len(P))] # on prend la liste des X des points de controles

    Py=[P[x][1] for x in range(len(P))] # on prend la liste des Y des points de controles

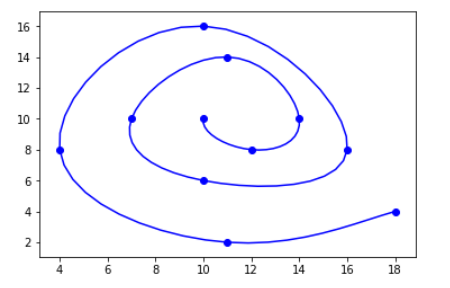
    xspline=[bcurve(i,u,k,Px) for i in np.arange(0,10,0.1)] #on lisse la courbe des points de controle

    yspline=[bcurve(i,u,k,Py) for i in np.arange(0,10,0.1)]

    plt.plot(xspline,yspline,'blue') # on affiche la courbe

    plt.scatter(x\_points\_figure,y\_points\_figure,c='blue') # on affiche les points

    plt.show()

affichage spirale :