```
"""Représentation plan
Auteur : Armand Caillon"""

from Polynomes import *
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
import math


def calc_z_coin (t,f, pasx,pasy) :
    """On initialise la surface à partir d'un coin"""
    m = len(t)
    n = len(t[0])
    Z = [[0 for _ in range (n)] for _ in range (m)]
    for i in range (1,n):
        X,Y = t[0][i-1]
        Z[0][i]=Z[0][i-1]-X/f*pasx
    for i in range (1,m):
        X,Y = t[i-1][0]
        Z[i][0]=Z[i-1][0]-Y/f*pasy
    for i in range (1,m):
        for j in range (1,n):
            X1 , Y1 = t[i][j-1]
            X2 , Y2 = t[i-1][j]
            Z[i][j] = 0.5*(Z[i][j-1]-X1/f*pasx + Z[i-1][j]-Y2/f*pasy)
    return Z

def calc_z_centre (t,f, pasx,pasy) :
    """On initialise à partir du centre"""
    m = len(t)
    n = len(t[0])
    Z = [[0 for _ in range (n)] for _ in range (m)]
    if m  % 2 ==0 :
        m1=m//2
    else:
        m1=m//2 + 1
    if n%2 == 0:
        n1=n//2
    else :
        n1=n//2 + 1
    for i in range (n1,n):
        X,Y = t[m1][i-1]
        Z[m1][i]=Z[m1][i-1]-X/f*pasx
    for i in range (n1-1,-1,-1):
        X,Y = t[m1][i+1]
        Z[m1][i]=Z[m1][i+1]+X/f*pasx
    for i in range (m1,m):
        X,Y = t[i-1][n1]
        Z[i][n1]=Z[i-1][n1]-Y/f*pasy
    for i in range (m1,-1,-1):
        X,Y = t[i+1][n1]
        Z[i][n1]=Z[i+1][n1]+Y/f*pasy
    for i in range (m1,m):
        for j in range (n1,n):
            X1,Y1=t[i][j-1]
            X2, Y2 = t[i-1][j]
            Z[i][j] = 0.5*(Z[i][j-1]-X1/f*pasx + Z[i-1][j]-Y2/f*pasy)
    for i in range (m1,m):
```

```
        for j in range (n1-1,-1,-1):
            X1,Y1=t[i][j+1]
            X2, Y2 = t[i-1][j]
            Z[i][j] = 0.5*(Z[i][j+1]+X1/f*pasx + Z[i-1][j]-Y2/f*pasy)
    for i in range (m1-1,-1,-1):
        for j in range (n1,n):
            X1,Y1=t[i][j-1]
            X2, Y2 = t[i+1][j]
            Z[i][j] = 0.5*(Z[i][j-1]-X1/f*pasx + Z[i+1][j]+Y2/f*pasy)
    for i in range (m1-1,-1,-1):
        for j in range (n1-1,-1,-1):
            X1,Y1=t[i][j+1]
            X2, Y2 = t[i+1][j]
            Z[i][j] = 0.5*(Z[i][j+1]+X1/f*pasx + Z[i+1][j]+Y2/f*pasy)
    return Z

t=[[(0,0),(-1,-1),(0,1),(-1,1),(-1,0),(1,1),(-1,0),(-1,1),(-1,0),(-1,1),(-1,1),(-1,2),
(0,0)],
   [(2,1),(1,1),(0,0),(-1,0),(-2,0),(-1,0),(0,0),(0,0),(0,-1),(-1,1),(-1,1),(-2,1),(-2,
1)],
   [(1,0),(0,0),(0,1),(-2,0),(-2,-1),(-1,2),(0,1),(0,0),(-1,1),(0,-1),(0,1),(-1,1),(-1,
3)],
   [(1,0),(0,0),(0,0),(-2,-1),(0,-1),(1,-1),(0,-1),(0,-1),(0,0),(0,-1),(-2,-1),(-1,0),
(-1,0)],
   [(2,0),(0,-1),(0,0),(-2,0),(-1,0),(0,0),(1,0),(1,0),(-1,-2),(0,0),(0,0),(-1,-1),(-1,
0)],
   [(1,0),(0,0),(1,0),(-1,0),(0,0),(0,0),(0,0),(0,-1),(0,0),(0,0),(2,0),(-1,1),(-1,-
1)],
   [(1,0),(0,0),(2,0),(-1,0),(0,0),(0,0),(-2,2),(0,0),(0,0),(-1,0),(-1,1),(0,0),(-1,-
1)],
   [(2,-2),(0,0),(1,-1),(0,-1),(0,0),(0,0),(0,0),(0,0),(-1,0),(0,0),(0,-1),(-2,0),(-1,-
1)],
   [(1,-1),(-1,1),(1,0),(-1,0),(0,0),(0,0),(0,-1),(0,0),(0,0),(0,0),(-1,0),(-1,-1),(-2,
-1)],
   [(0,0),(1,-1),(-1,0),(2,0),(0,0),(0,-1),(0,0),(0,0),(0,0),(0,0),(0,-1),(0,0),(0,0)],
   [(0,0),(1,-1),(-1,0),(1,-2),(0,-1),(1,-1),(0,0),(0,-1),(0,-1),(-1,-1),(0,0),(0,0),
(0,0)]]


plan_z1=calc_z_centre(t,200,5,5)
n1=len(plan_z1[0])
m1=len(plan_z1)
lx=[i*0.5 for i in range (m1)]
ly=[j*0.5 for j in range (n1)]


z1=interpole2v(lx,ly,plan_z1)

def plan1 (x,y):
    val=0
    n=len(z1)
    m=len(z1[0])
    for i in range (n):
        for j in range (m):
            val += x**i * y**j * z1[i][j]
    return val

def plan1_dec(x,y):
    return plan1(3/2*x+2.5,3/2*y+2.5)
```

```
def plan1_polaire_decale(rho,phi):
    return plan1(3/2*rho*math.sin(phi)+2.5,3/2*rho*math.cos(phi)+2.5)

def plan2 (x,y):
    val=0
    n=len(z2)
    m=len(z2[0])
    for i in range (n):
        for j in range (m):
            val += x**i * y**j * z2[i][j]
    return val

ax = Axes3D(plt.figure())
plan1_dec = np.vectorize(plan1_dec)
R = np.arange(0,1,0.05)
Phi = np.arange(0,2*math.pi+0.1,0.05)
R,P = np.meshgrid(R, Phi)
X , Y = R*np.cos(P) , R*np.sin(P)
plan1_polaire_decale = np.vectorize(plan1_polaire_decale)
Z = plan1_dec(X, Y)
plt.xlabel('x',color = 'red')
plt.ylabel('y', color = 'red')
ax.plot_surface(X, Y, Z, cmap=cm.coolwarm)
plt.show()

plan_z2 = calc_z_centre(t, 200, 5, 5)
n2 = len(plan_z2[0])
m2 = len(plan_z2)
lx = [i * 0.5 for i in range(m2)]
ly = [j * 0.5 for j in range(n2)]

z2 = interpole2v(lx, ly, plan_z2)

ax = Axes3D(plt.figure())
plan2=np.vectorize(plan2)
X = np.arange(1, 4, 0.1)
Y = np.arange(1, 4, 0.1)
X, Y = np.meshgrid(X, Y)
Z = plan2(X , Y)
ax.plot_surface(X, Y, Z,cmap = cm.coolwarm)
plt.show()
```