```
""" Projection Zernike sur plusieurs ordres
Auteur : Lucas Barbier"""

from scipy.integrate import dblquad
from Zernike import *
from Représentation_plan import *


def ps(f,g,domaine_rho,domaine_phi):
    """produit scalaire"""
    return dblquad(lambda rho, phi : 1/(math.pi)*f(rho,phi)*g(rho,phi)*rho,
domaine_phi[0], domaine_phi[1], lambda rho : domaine_rho[0],
lambda rho : domaine_rho[1], epsabs=1e-5, epsrel=1e-5)[0]

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

for i in  range (6):
    """On va afficher la somme des polynomes de zernike coefficientés jusqu'à l'ordre
i"""

    z1=interpole2v(lx,ly,plan_z1)
    projection_zernike = []
    for n in range(i+6):
        pr = []
        for m in range(-n,n+1,2):
            Zernike_xy(0,0,m,n) #Calcul du polynôme Z(n,m)
            prod = ps(Zernike,plan1_polaire_decale,[0,1],[0,2*math.pi])/ps(Zernike,
Zernike,[0,1],[0,2*math.pi]) #Projection du front d'onde sur Z(n,m)
            pr.append(prod)
        projection_zernike.append(pr)

    def resultat(x,y):
        res = 0
        for n in range(i+6):
            k = 0
            for m in range(-n,n+1,2):
                Zernike_xy.n, Zernike_xy.m = n,m
                res += (projection_zernike[n][k])*Zernike_xy(x,y)
                k += 1
        return res

    def erreur_ordre():
        e = []
        for n in range(i+6):
            s = 0
            r = 0
            for m in range(-n,n+1,2):
                for l in range(len(t)):
                    for j in range(len(t[0])):
                        x,y = 0.5*l,0.5*j
                        if 2/3*math.sqrt(x*x+y*y) <= 1:
                            r += 1
                            s += (resultat(x,y)-plan1(x,y))**2
                s /= r
            e.append(s)
        return e

    plt.close()
plt.plot([0,1,2,3,4,5,6,7,8,9,10],erreur_ordre())
plt.xlabel('ordre')
plt.ylabel('ecart quadratique')
plt.show
ax = Axes3D(plt.figure())
resultat = np.vectorize(resultat)
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)
Z = resultat(X,Y)
ax.plot_surface(Y, X, Z, cmap=cm.coolwarm)
plt.xlabel('x', color = 'red')
plt.ylabel('y', color = 'red')
plt.show()
```