```
# -*- coding: utf-8 -*-
# Fichier tp3.py

from numpy import * # importation du module numpy
from numpy.linalg import *  # importation du module numpy.linalg
from matplotlib.pyplot import * # importation du module matplotlib.pyplot
from mpl_toolkits.mplot3d import Axes3D # importation du module mpl_toolkits.mplot3d
import time
from pylab import *


# Fonction définissant U0(x)
def U0(x):
    y = zeros(shape(x))
    for i in arange(0,size(x),1):
         if (x[i] >= 0.4 and x[i] < 0.5):
              y[i] = 10*(x[i]-0.4)
         elif (x[i] >= 0.5 and x[i]<= 0.6):
              y[i] = 10*(0.6-x[i])
         else:
              y[i] = 0
    return y

# Fonction définissant la solution exacte de l'équation
#attention ct = c*t
def solex(x,ct):
    y = zeros(shape(x))
    y = U0(x-ct)
    return y

print("Choix du schéma pour calcul des U(j): ")
print("1- schéma décentré à gauche")
print("2- schéma décentré à droite")
print("3- schéma de Lax-Friedrichs")
meth = int(input('Choix = '))

print('Choix de la vitesse de transport c')
c = float(input('c = '))

print('Choix du nombre Ns de points interieurs du maillage')
Ns = int(input('Ns = '))

h = 1./(Ns + 1.)
print('h = ' , h)



print('Choix du pas dt en temps')
dt = float(input('dt = '))


print('c*dt/h* = ' , c*dt/h)

print('Choix du temps final T')
T = float(input('T = '))

# Maillage

h = 1./(Ns + 1.)
X = linspace(0.,1.,Ns+1)
Xh = X[0:Ns+1]
M = int((T/dt) - 1)
```

```
CFL=dt/h


# init Uh avec la condition T=0
Uh = U0(Xh)

#array pr T=T+1
Uhn = zeros(shape(Uh))

# Calcul iteratif des vecteurs U(j) par le schéma choisi

# Erreur,Erreur temporaire
Err = 0
Errn = 0


#Animation
#ion()
#line1,  = plot(linspace(0,1,100), solex(linspace(0,1,100),0), label = 'sol exacte')
#line2, = plot(Xh, Uh, label = 'sol approchee')

#xlabel('X')
#ylabel('Y')
#legend()

for j in arange(1, M+1):
    for i in arange(1,Ns):
        if(meth == 1):
            Uhn[i] =  Uh[i] - c*CFL*(Uh[i] - Uh[i-1])
        elif(meth == 2):
            Uhn[i] =  Uh[i] - c*CFL*(Uh[i+1] - Uh[i])
        elif(meth == 3):
            Uhn[i] =  .5*(Uh[i-1] + Uh[i+1]) - .5*c*CFL*(Uh[i+1] - Uh[i-1])
    # Evaluation de la solution exacte au temps j*dt
    U=solex(X,j*dt*c)
    Errn = amax(absolute(U - Uhn))
    if (Err < Errn):
      Err = Errn
    #Affichage mise a jour
    #line1.set_ydata(solex(linspace(0,1,100),j*dt*c))
    #line2.set_ydata(Uh)
    #draw()
    #pause(.01)
    # swap array
    Uh, Uhn = Uhn, Uh

# Tracé du graphe de la fonction Uh(x,T)
# Tracé du graphe de la fonction U(x,T)

plot(linspace(0,1,100),solex(linspace(0,1,100),T*c), label = 'sol ex')
plot(Xh, Uh, label = 'sol approchee')
xlabel('X')
ylabel('Y')
legend()
#ioff()
show()
```