import numpy as np

from matplotlib import pyplot as pl

### Données

l = 1e-4 ; t0 = 0 ; tf = 50 ; T0 = 50 ; T1 = 40 ; T2 = 60 ; n = 100

hx=1/n

ht=tf/n

### Vecteur initial

# Propo 1 : méthode liste (voir Sup)

def Y0(n,T0,T1,T2): # tableau de longueur (n+1)

A=[[T1]]

for k in range(1,n): # (n-1) itérations

A.append([T0])

A.append([T2])

return np.array(A)

# Propo 2 : méthode liste en compréhension (voir Sup)

def Y0(n,T0,T1,T2):

A=[[T0] for k in range(n+1)]

A[0]=[T1]

A[-1]=[T2]

return np.array(A)

# Propo 3 : méthode avec initialisation d'un tableau de zéros (voir Spé)

def Y0(n,T0,T1,T2):

A=np.zeros((n+1,1)) # attention : doubles parenthèses

for k in range (1,len(A)-1):

A[k,0]=T0

A[0,0]=T1

A[-1,]=T2

return A

### Matrice thermique

def K(n):

mat = np.zeros((n+1,n+1))

for i in range(1,n): # (1,n) pour sauter les lignes extrêmes d'index 0 et n

mat[i,i-1] = 1

mat[i,i] = -2

mat[i,i+1] = 1

return l\*ht/hx\*\*2\*mat # multiplication de tous les termes de la matrice. Vive Numpy !

# Et aussi

def K(n):

mat = np.zeros((n+1,n+1))

for i in range(1,n):

mat[i-1:i+2] = [1,-2,1] # Slicing !

return l\*ht/hx\*\*2\*mat

# Plus long :

def K(n):

mat = np.zeros((n+1,n+1))

for i in range(1,n):

mat[i,i] = -2

for j in range(n+1):

if j==i-1 or j==i+1:

mat[i,j]=1

return l\*ht/hx\*\*2\*mat

### Vecteur Yj

def chaleur(j,n,T0,T1,T2):

Y=Y0(n,T0,T1,T2)

for k in range(j):

Y=Y+ np.dot(K(n),Y) # ou Y=Y+K(n)@Y

return Y

""" Remarque : en récursif,

if j==0:

return Y0(n,T0,T1,T2)

else:

c=chaleur(j-1,n,T0,T1,T2)

return c+ np.dot(K(n),c)

"""

# Nécessite au maximum O(n\*\*3) multiplications !!!

### Couple (X,Y) :

def coupleXY(j,n,T0,T1,T2):

X=np.arange(0,1+1/n,1/n) # ou np.linspace(0,1,n+1)

Y=[x for x in chaleur(j,n,T0,T1,T2)]

# np.transpose(chaleur(j,n)) ne marche pas car renvoie une liste de liste, alors qu'on veut une liste. Marcherait : np.transpose(chaleur(j,n))[0]

return X,Y

### 7.a) Représentation statique

n=100

pl.figure()

for j in range(n+1):

Lx,Ly=coupleXY(j,n,T0,T1,T2)

pl.plot(Lx,Ly)

pl.show()

### 7.b) Représentation graphique animée

n=100

from matplotlib import animation

# First set up the figure, the axis, and the plot element we want to animate

fig = pl.figure()

ax = pl.axes(xlim=(0, 1), ylim=(min(T0,T1,T2), max(T0,T1,T2)))

dot, = ax.plot([0], [0], 'r')

# initialization function: plot the background of each frame

def init():

return dot,

# animation function. This is called sequentially

def animate(j):

new\_x, new\_y = coupleXY(j,n,T0,T1,T2)

dot.set\_data(new\_x, new\_y)

return dot,

# call the animator. blit=True means only re-draw the parts that have changed.

anim = animation.FuncAnimation(fig, animate, init\_func=init, frames=n, blit=True, interval=100, repeat=False)

pl.show()