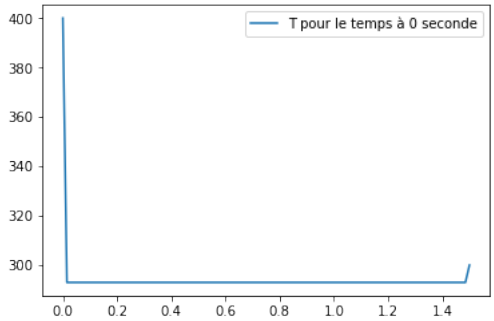
équation de la chaleur 1D: Méthode Implicite

Ce script permet de calculer l'évolution de la température dans un barreau calorifugé latéralement,
maintenu à des températures différentes à chacune de ses extrémités.

```
#Importer les bibliothèques python utiles
import numpy as np
import scipy as sc
import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D # Fonction qui permet de tracer en
3D la temperature
# Fonction de création d'une matrice diagonale avec pour donnée d'entrée c0
= nbre de points
# C2 valeur de la diagonale centrale
# C1 valeur de la diagonale autour de la diagonale centrale
# appeler les fonctions juste après avoir appelé les bibliothèques

def MatDiag(c0,c1,c2): # c0 correspond aux nbres de lignes dans la matrice,
nbre d'équations
    M = np.zeros ((c0+1,c0+1),float)
    M[0,0]=M[c0,c0]=1
    for r in range (1,c0):
        M[r,r]=c2
        M[r,r-1]=M[r,r+1]=c1
    return M
#Les données d'entrée de notre probleme
alpha=20e-6        # diffusivité thermique en m2.s-1
L=1.5              # longueur de la barre en m
tf=float(2*60*60)  # temps d'évolution de 2 heures convertis en secondes de
type float (réel)
Tg=400.            # soit on met un point . pour préciser que c'est un réel
soit on précise float(400)
Td=float(300)     # temperature de droite idem de type float
Nx=100            # nbre de points de calcul que l'on choisit sur le
barreau
dt=8.             # pas de temps qui est un réel
Nt=int(tf/dt)     # nbre de points de calcul dans le temps
dx=L/Nx           # nbre de pas d'espace
K=alpha*dt/(dx*dx)# constante K
x=np.linspace(0,L,num=Nx+1) # créer tableau allant de 0 à L avec Nx+1
valeurs dans le tableau, Nx+1 car on compte depuis 0
# Vérification de la condition de stabilité CFL
CFL=alpha*dt/(dx*dx)
print ("CFL = ",CFL)
if CFL<=0.5:
    print ("La condition CFL est respectée. La méthode explicite est
stable")
else:
    print ("La condition CFL n'est pas respectée. La méthode explicite est
instable.")
CFL =  0.7111111111111111
La condition CFL n'est pas respectée. La méthode explicite est instable.
whos
Variable   Type        Data/Info
--------------------------------
Axes3D     type        <class 'mpl_toolkits.mplot3d.axes3d.Axes3D'>
CFL        float       0.7111111111111111
K          float       0.7111111111111111
L          float       1.5
```

```
MatDiag     function    <function MatDiag at 0x7f06f1a29510>
Nt          int         900
Nx          int         100
Td          float       300.0
Tg          float       400.0
alpha       float       2e-05
dt          float       8.0
dx          float       0.015
mpl         module      <module 'matplotlib'
from<...>/matplotlib/__init__.py'>
np          module      <module 'numpy' from
'/op<...>kages/numpy/__init__.py'>
plt         module      <module
'matplotlib.pyplo<...>es/matplotlib/pyplot.py'>
sc          module      <module 'scipy' from
'/op<...>kages/scipy/__init__.py'>
tf          float       7200.0
x           ndarray     101: 101 elems, type `float64`, 808 bytes
# Conditions initiales, on crée un tableau pour que toute la barre soit à
293 K puis temp. aux extrémités
T=293*np.ones((Nx+1,Nt+1)) # on crée un tableau T de température rempli de
1 que l'on multiplie par 293 K
u=293*np.ones((Nx+1,Nt+1))
for i in range(0,Nx+1): # autre manière de d'affecter 293 sur toute la
longueur
    T[i,0]=293.  # attention ajouter l'indentation pour que le programme
comprenne la boucle for
                 # module = bibliothèque de Python
# Conditions aux limites
for n in range (0,Nt+1):
    T[0,n]=Tg  # On affecte à gauche de la barre la température Tg
    T[Nx,n]=Td # On affecte à droite de la barre la température Td
# Profil de température
plt.plot (x,T[:,0]) # en abscisse la longueur, en ordonnée peut importe la
position en x, au temps t=0
plt.legend (['T pour le temps à 0 seconde'])
<matplotlib.legend.Legend at 0x7f06f19912e8>
```




```
# Etape 6: calcul de la température à tous les Nt points et tous les noeuds
i du barreau
A=MatDiag (Nx,-K,1+2*K) # Nx nbre de points dans l'analyse, -K ce qui
entoure la diag, 1+2*K la diago
print (A)
[[ 1.          0.          0.         ...  0.          0.
   0.        ]
 [-0.71111111  2.42222222 -0.71111111 ...  0.          0.
   0.        ]
 [ 0.         -0.71111111  2.42222222 ...  0.          0.
   0.        ]
 ...
 [ 0.          0.          0.         ...  2.42222222 -0.71111111
```

```
   0.        ]
 [ 0.          0.          0.         ... -0.71111111  2.42222222
  -0.71111111]
 [ 0.          0.          0.         ...  0.          0.
   1.        ]]
# Inversion de la matrice A
AI = np.linalg.inv(A)
# Résolution du système à chaque pas de temps
for n in range(0,Nt):
    T[:,n+1]=np.dot(AI,T[:,n]) # toutes les valeurs du barreau en N+1 en
faisant le produit scalaire

# Etape 7: tracer l'évolution de la température dans le barreau
plt.plot(x,T[:,0],'or', x,T[:,int(Nt/2)],'--k', x,T[:,Nt],'-
b',x,u[:,int(Nt/2)],'+g')
plt.legend(['T pour t=0s', 'T pour t=1 h', 'T pour t=2 h'])
plt.xlabel('Longueur (m)')
plt.ylabel ('Température T(K)')

fig=plt.figure()
ax=fig.add_subplot(111,projection='3d')

x_m,tps_m=np.meshgrid(x,tps)
Temp=ax.plot_surface(x_m,tps_m,np.transpose(T),cmap='jet')
plt.xlabel('Longueur (m)')
plt.ylabel ('Temps (s)')
fig.colorbar(Temp, skrink=0.5, aspect=5)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-12-c18cbd5bee08> in <module>
      9
     10 #x_m,tps_m=np.meshgrid(x,tps)
---> 11 Temp=ax.plot_surface(x_m,tps_m,np.transpose(T),cmap='jet')
     12 plt.xlabel('Longueur (m)')
     13 plt.ylabel ('Temps (s)')

NameError: name 'x_m' is not defined
```