

```

import numpy as np                # Pour les listes
import time as t                  # Pour calculer la
durée d'exécution
from matplotlib import pyplot as plt  # Pour le tracé

"""
Created on Thu May 23 10:27:20 2013

@author: eric ouvrard - PC Dupuy de Lôme - Lorient
site: http://cpgedupuydelome.fr/spip.php?page=exercices
"""

def calc_norme(vecteur):
    N=len(vecteur)
    norme=0
    for i in range(N):
        norme=norme+(vecteur[i])**2
    norme=(norme)**0.5
    return norme

def schema_explicite(T0,ItMax):
    debut=t.time()
    N=len(T0)
    # nombre de lignes
    T_tous_k=np.zeros((N,ItMax))
    # On définit la matrice
    for i in range(0,N):
        # On remplit la
        premiere colonne de la matrice
        T_tous_k[i,0]=T0[i]

    T_tous_k[0,1]=r*Tint+(1-2*r)*T_tous_k[0,0]+r*T_tous_k[
1,0]
    for i in range(1,N-1):
        T_tous_k[i,1]=r*T_tous_k[i-1,0]+(1-2*r)*T_tous_k[i
,0]+r*T_tous_k[i+1,0]
        T_tous_k[N-1,1]=r*T_tous_k[N-2,0]+(1-2*r)*T_tous_k[N-1
,0]+r*Text2
    test=calc_norme(T_tous_k[:,1]-T_tous_k[:,0])
    j=2
    while (test>.005 and j<ItMax-1):
        T_tous_k[0,j]=r*Tint+(1-2*r)*T_tous_k[0,j-1]+r*T_t
ous_k[1,j-1]
        for i in range(1,N-1):
            T_tous_k[i,j]=r*T_tous_k[i-1,j-1]+(1-2*r)*T_to
us_k[i,j-1]+r*T_tous_k[i+1,j-1]
            T_tous_k[N-1,j]=r*T_tous_k[N-2,j-1]+(1-2*r)*T_tous

```

```

_k[N-1,j-1]+r*Text2
    test=calc_norme(T_tous_k[:,j]-T_tous_k[:,j-1])
    j=j+1
fin=t.time()
duree=fin-debut
return [j,T_tous_k,duree]

def CalcTkpl(M,d):
    cprim=np.zeros(len(d))
    dprim=np.zeros(len(d))
    cprim[0]=M[1,0]/M[0,0]
    dprim[0]=d[0]/M[0,0]
    for k in range(1,len(d)-1):
        cprim[k]=M[k+1,k]/(M[k,k]-M[k-1,k]*cprim[k-1])
    for i in range(1,len(d)):
        dprim[i]=(d[i]-M[i-1,i]*dprim[i-1])/(M[i,i]-
M[i-1,i]*cprim[i-1])
    u=np.zeros(len(d))
    u[-1]=dprim[-1]
    for i in reversed(range(0,len(d)-1)):
        u[i]=dprim[i]-cprim[i]*u[i+1]
    return u

def schema_implicit(T0,ItMax):
    debut=t.time()
    N=len(T0)
    # nombre de lignes
    T_tous_k=np.zeros((N,ItMax))
    # On définit la matrice
    for i in range(0,N):
        # On remplit la
        premiere colonne de la matrice
        T_tous_k[i,0]=T0[i]
        #On construit la matrice M
        M=np.zeros((N,N))
        M[0,0]=1+2*r
        M[0,1]=-r
        M[-1,-1]=1+2*r
        M[-1,-2]=-r
        for i in range(1,N-1):
            M[i,i-1]=-r
            M[i,i]=1+2*r
            M[i,i+1]=-r
        #On construit le vecteur colonne v
        v=np.zeros(N)
        v[0]=Tint
        v[-1]=Text2

```

```

#Profil de temperature pour k=1
profil=CalcTkpl(M,T0+r*v)
#On affecte à la deuxieme colonne de T_tous_k
T_tous_k[:,1]=profil
#On écrit une boucle pour compléter les colonnes
suivantes
test=calc_norme(T_tous_k[:,1]-T_tous_k[:,0])
j=2
while (test>.005 and j<ItMax-1):
    T_tous_k[:,j]=CalcTkpl(M,T_tous_k[:,j-1]+r*v)
    test=calc_norme(T_tous_k[:,j]-T_tous_k[:,j-1])
    j=j+1
fin=t.time()
duree=fin-debut
return [j,T_tous_k,duree]

```

```

#Début du programme principal
epais,conduc,rho,cp=0.4,1.65,2156,1000
Tint,Text1,Text2=20,10,-10
N,Dt=60,25

```

```

#Calcul des coefficients a et b
a=(Text1-Tint)/epais
b=Tint

```

```

#Construction du vecteur x d'après la définition du I1B
de x(1) à x(N)... ne doit pas apparaitre dans cette liste
x(0) et x(N+1)!
dx=epais/N
x=np.linspace(dx,epais-dx,N) #avec linspace, epais-dx est
bien la dernière valeur de la liste constituée

```

```

# liste des températures initiales
T0=np.zeros(N)
T0=a*x+b #N'est possible que sur des listes numpy,
sinon réaliser une boucle

```

```

# Expression de alpha et r
alpha=rho*cp/conduc
r=Dt/(dx**2*alpha)

```

```

choix=input("Pour la méthode implicite tapez 1, sinon
tapez 2")

if choix == '1':
    res=schema_implicite(T0,10000)
    temp=res[1]
    print("Il y a eu %s itérations et une durée de
résolution %.3f" % (res[0],res[2]))
    for p in np.arange(0,res[0],res[0]/100):
        plt.plot(x,temp[:,p])

elif choix=='2':
    res=schema_explicite(T0,10000)
    temp=res[1]
    print("Il y a eu %s itérations et une durée de
résolution %.3f" % (res[0],res[2]))
    for p in np.arange(0,res[0],res[0]/100):
        plt.plot(x,temp[:,p])

else:
    print("Suivez bien la consigne!")

```