```
import math
import numpy as np
import matplotlib.pyplot as plt

#u,t = -V u,x + k u,xx  -lamda u + f

# PHYSICAL PARAMETERS
K = 0.1     #Diffusion coefficient
L = 1.0     #Domain size
Time = 20.  #Integration time


V=1
lamda=1

# NUMERICAL PARAMETERS
NX = 2  #Number of grid points
NT = 10000   #Number of time steps max
ifre=1000000  #plot every ifre time iterations
eps=0.001     #relative convergence ratio
niter_refinement=30      #niter different calculations with variable mesh size

error=np.zeros((niter_refinement))
errorh1=np.zeros((niter_refinement))
itertab=np.zeros((niter_refinement))
normTexx=np.zeros((niter_refinement))

for iter in range (niter_refinement):
    NX=NX+3

    dx = L/(NX-1)                 #Grid step (space)
    dt = dx**2/(V*dx+K+dx**2)   #Grid step (time)  condition CFL de stabilite 10.4.5
    print(dx,dt)
    itertab[iter]=dx

    ### MAIN PROGRAM ###

    # Initialisation
    x = np.linspace(0.0,1.0,NX)
    T = np.zeros((NX)) #np.sin(2*np.pi*x)
    F = np.zeros((NX))
    rest = []
    RHS = np.zeros((NX))

    Tex = np.zeros((NX)) #np.sin(2*np.pi*x)
    Texx = np.zeros((NX)) #np.sin(2*np.pi*x)
    for j in range (1,NX-1):
        Tex[j] = np.sin(2*j*math.pi/NX)
    for j in range (1,NX-1):
        Texx[j]=(Tex[j+1]-Tex[j-1])/(2*dx)  #np.cos(j*math.pi/NX)*math.pi/NX
        normTexx[iter]+=Texx[j]**2
        Txx=(Tex[j+1]-2*Tex[j]+Tex[j-1])/(dx**2)  #-np.sin(j*math.pi/NX)*(math.pi/NX)**2    #
        F[j]=V*Texx[j]-K*Txx+lamda*Tex[j]


    dt = dx**2/(V*dx+2*K+abs(np.max(F))*dx**2)   #Grid step (time)  condition CFL de stabilite 10.4.5

    plt.figure(1)


    # Main loop en temps
```

```
    #for n in range(0,NT):
    n=0
    res=1
    res0=1
    while(n<NT and res/res0>eps):
        n+=1
    #discretization of the advection/diffusion/reaction/source equation
        res=0
        for j in range (1, NX-1):
            xnu=K+0.5*dx*abs(V)
            Tx=(T[j+1]-T[j-1])/(2*dx)
            Txx=(T[j-1]-2*T[j]+T[j+1])/(dx**2)
            RHS[j] = dt*(-V*Tx+xnu*Txx-lamda*T[j]+F[j])
            res+=abs(RHS[j])

        for j in range (1, NX-1):
            T[j] += RHS[j]
            RHS[j]=0


        if (n == 1 ):
            res0=res

        rest.append(res)
    #Plot every ifre time steps
        if (n%ifre == 0 or (res/res0)<eps):
            print(n,res)
            plotlabel = "t = %1.2f" %(n * dt)
            plt.plot(x,T, label=plotlabel,color = plt.get_cmap('copper')(float(n)/NT))


    print(n,res)
    plt.plot(x,T)

    plt.xlabel(u'$x$', fontsize=26)
    plt.ylabel(u'$T$', fontsize=26, rotation=0)
    plt.title(u'ADRS 1D')
    plt.legend()

    plt.figure(2)
    plt.plot(np.log10(rest/rest[0]))

    err=np.dot(T-Tex,T-Tex)
    errh1=0
    for j in range (1,NX-1):
        errh1+=(Texx[j]-(T[j+1]-T[j-1])/(2*dx))**2

    error[iter]=np.sqrt(err)
    errorh1[iter]=np.sqrt(err+errh1)
    print('norm error=',error[iter],errorh1[iter])


# plt.figure(3)
# plt.plot(x,Tex, label=plotlabel,color = plt.get_cmap('copper')(float(n)/NT))



plt.figure(3)

plt.subplot(1, 2, 1) # row 1, col 2 index 1
plt.plot(itertab,error)
# plt.plot(itertab,0.3+20*itertab**1.5)
```

```
# plt.plot(itertab,0.3+20*itertab**2)
# plt.plot(itertab,0.3+20*itertab)
plt.title("L2 error")
plt.xlabel('Refinement iterations')
plt.ylabel('Error L2')

plt.subplot(1, 2, 2) # index 2
plt.plot(itertab,errorh1)
plt.title("H1 Error")
plt.xlabel('Refinement iteration')
plt.ylabel('Error H1')

plt.show()

#%%
# Define the Gaussian function
def target_func(x, A, B):
    #y = A*np.exp(-1*B*x**2)
    y=A*x**B
    return y

from scipy.optimize import curve_fit

normTexx=np.sqrt(normTexx)

xdata=itertab[2:]
ydataL2=error[2:]/normTexx[2:]
parametersL2, covariance = curve_fit(target_func, xdata, ydataL2)
xdata=itertab[2:]
ydataH1=errorh1[2:]/normTexx[2:]
parametersH1, covariance = curve_fit(target_func, xdata, ydataH1)
print(parametersL2)
print(parametersH1)

plt.subplot(1, 2, 1) # row 1, col 2 index 1
plt.plot(xdata,ydataL2)
plt.plot(xdata,target_func(xdata, parametersL2[0], parametersL2[1]),'red')
plt.plot(xdata,target_func(xdata, parametersL2[0], parametersL2[1]))
plt.title("L2 error")
plt.xlabel('Refinement iterations')
plt.ylabel('Error L2')

plt.subplot(1, 2, 2) # index 2
plt.plot(xdata,ydataH1)
plt.plot(xdata,target_func(xdata, parametersH1[0], parametersH1[1]))
plt.title("H1 Error")
plt.xlabel('Refinement iteration')
plt.ylabel('Error H1')

plt.show()

#%%
#export/import donnÃ©es

import pandas as pd
arr = np.asarray([ xdata, ydataL2, ydataH1 ])
pd.DataFrame(arr).to_csv('data.csv',header=None)

df = pd.read_csv('data.csv',header=None)
df=np.array(df)
```