Faculté des sciences de Montpellier
Rapport séance 2

Étudiant: Giscard Leonel Zouakeu

Probleme:

$$\begin{cases} \partial_t u(t,s) + V(t,s)\partial_s u(t,s) - \nu\partial_s^2 u(t,s) + \lambda u(t,s) = f(t,s) & (t,s) \in (0,L).(0,+\infty) \\ V(t,s) = 1, \nu = 0.01, \lambda = 1, L = 1 \end{cases}$$

La solution exacte de ce probleme est:

$$u_{ex}(t,s) = exp(-10(s-L/2)^2))$$

Euleur explicite

$$u_i^{n+1} = au_{i+1}^n + bu_i^n + cu_{i-1}^n + f_i^n$$

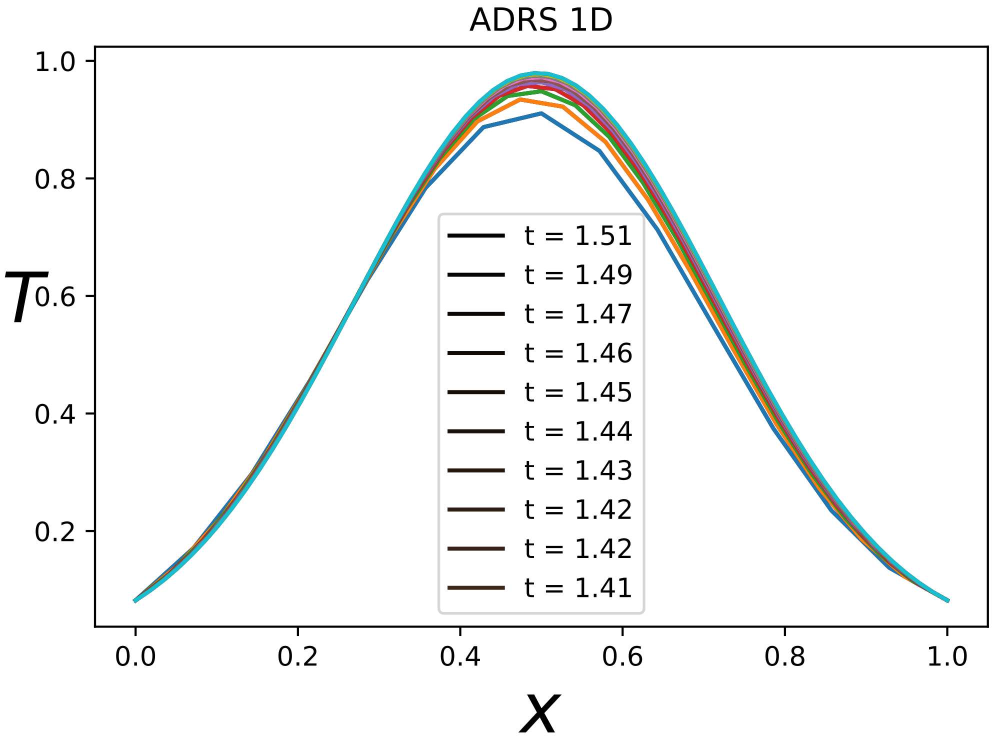
Avec : $a = dt(1/h^2 - 1/h), b = dt(-2/h^2 + 1/h - 1) + 1, c = dt/h^2$

compréhension et modification du code adrs.py

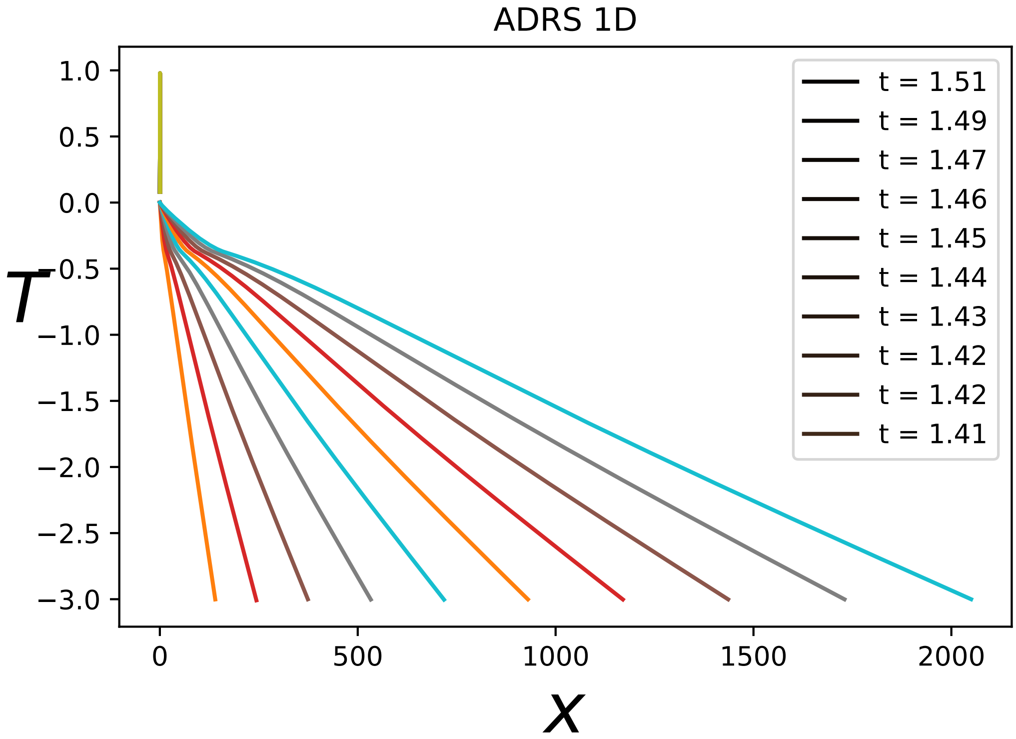
```python
import math
import numpy as np
import matplotlib.pyplot as plt

#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 = 10    #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=10      #niter different calculations with variable mesh size

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

for iter in range (niter_refinement):
  NX=NX+5

  dx = L/(NX-1)                 #Grid step (space)
  dt = dx**2/(V*dx+4*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.exp(-10*((x-L/2))**2)
F = np.zeros((NX))
rest = []
RHS = np.zeros((NX))

Tex = np.exp(-10*((x-L/2))**2)
#for j in range (1,NX-1):
  #Tex[j] = 1
for j in range (1,NX-1):
  Tx=(Tex[j+1]-Tex[j-1])/(2*dx)
  Txx=(Tex[j+1]-2*Tex[j]+Tex[j-1])/(dx**2)
  F[j]=V*Tx-K*Txx+lamda*Tex[j]


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.sqrt(np.dot(T-Tex,T-Tex))
  error[iter]=err
  print('norm error=',err)


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

plt.figure(3)
plt.plot(itertab,error)

plt.show()
```

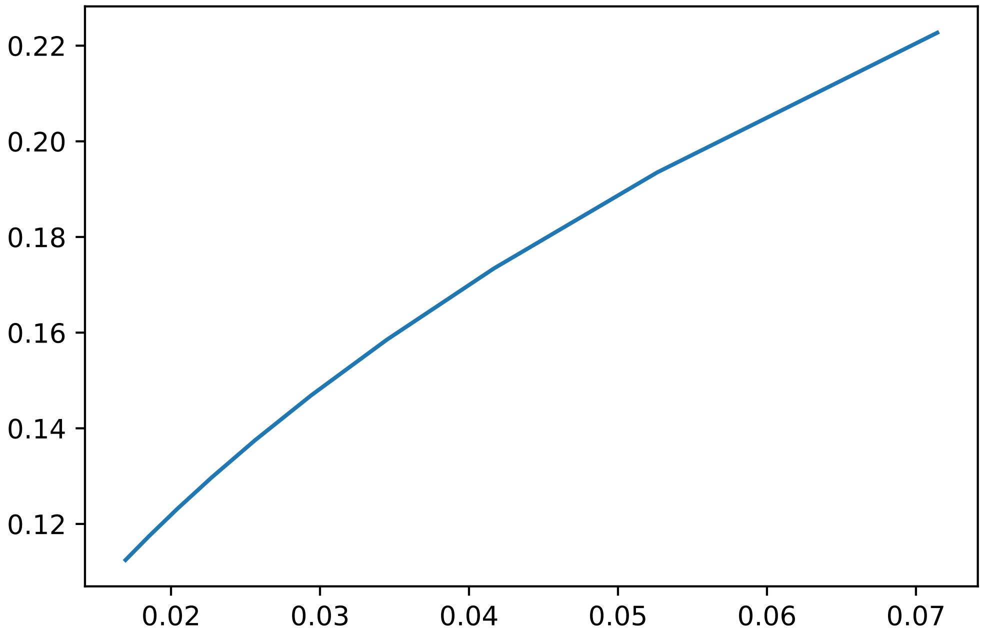


ADRS 1D
1.0
0.8
0.6
0.4
0.2
T
0.0
0.2
0.4
0.6
0.8
1.0
x
t = 1.51
t = 1.49
t = 1.47
t = 1.46
t = 1.45
t = 1.44
t = 1.43
t = 1.42
t = 1.42
t = 1.41



Erreur

Dans le cas stationnaire , on cherche à faire la courbe de l'erreur:
$||u_{ex} - u_{appro}||_{L^2}$


0.22
0.20
0.18
0.16
0.14
0.12
0.02
0.03
0.04
0.05
0.06
0.07