Projet 6, Modélisation d'une ventilation

Zouiche Omar et David Czarnecki

21 Mars 2021

Nous allons migrer le code du projet 3 de l'isolateur thermique afin de modéliser et dimensionner d'une ventillation pour le refroidissement d'un processeur, l'équation à résoudre est : $\forall x \in [0, L]$ et $\forall t \in [0, T]$:

$$\frac{\partial T}{\partial t} + V\frac{\partial T}{\partial x} - k\frac{\partial^2 T}{\partial x^2} = f(x_{proc})$$

Elle modélise le phénomène d'advection. La constante V est la vitesse de l'air soufflé dans le sens de x. $ $\partial T \frac{}{\partial x\text{\$estl'advectionet}\frac{\partial^2 T}{\partial x^2}}$ est la diffusion. Mais nous partons de l'équation de la chaleur :

$$\begin{cases} \frac{\partial T}{\partial t}(x,t) - k\frac{\partial^2 T}{\partial x^2}(x,t) = f(x) \\ u(x,0) = u_0(x) \end{cases}$$

On discrétise aussi les dérivée suivant la formule :

$$\frac{\partial T}{\partial t}(x_i, t_{n+1}) \simeq \frac{T(x_i, t_{n+1}) - T(x_i, t_n)}{dt}$$

et

$$\frac{\partial^2 T}{\partial x^2}(x_i, t_{n+1}) \simeq \frac{T(x_{i+1}, t_n) - 2T(x_i, t_n) + T(x_{x+1}, t_n}{dx^2}$$

On considère les fonctionnelles :

$$J_1(V) = max(0, T(V, x = 1/2) - 1.5)$$

et :

$$J_2(V) = \frac{1}{2}V^2$$

qui prennent la vitesse V comme variable. Elles nous seront utiles pour tracer le front de Pareto exprimé par $(J_1(V), J_2(V))$.

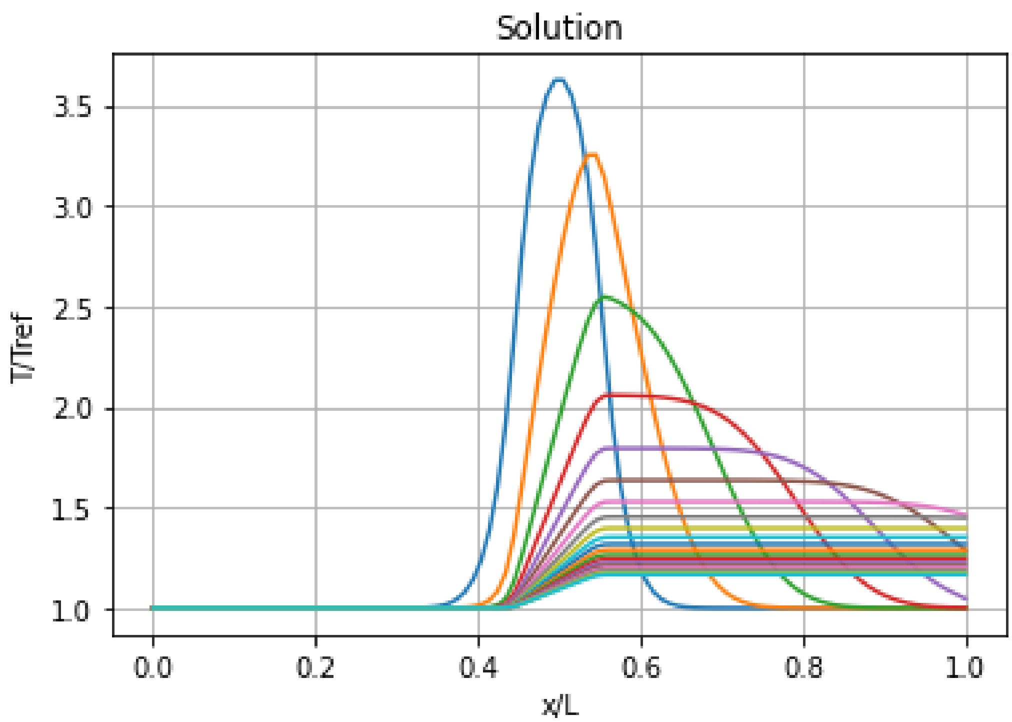
[14]:
```python
import numpy as np
import matplotlib.pyplot as plt

Fmax = 3   # max heat production source level
V = 0     #flow velocity in range [0, 2 m/s]
K = 0.001  #Reference Diffusion coefficient
L = 1.0    #Domain size
Time = 1   #Integration time
NX = 100   #Number of grid points
nbctrl=20  #Sampling of V range
```

```python
ifre = 1000
Tcible = 1.5
Tref = 1


# Initialisation
x = np.linspace(0.0,1.0, NX)
T = np.ones(NX)
RHS = np.zeros(NX)
F = np.zeros(NX)
cost = np.zeros(nbctrl)
cost1 = np.zeros(nbctrl)
cost2 = np.zeros(nbctrl)
ctrlsave = np.zeros(nbctrl)

# PARAMETRES
dx = L/(NX-1)

DCNTRL = 0.1
CNTRL = -DCNTRL
gradtarget = 0


for jctrl in range(0,nbctrl):
    CNTRL+= DCNTRL
    ctrlsave[jctrl]=CNTRL
    V = CNTRL
    print('V=',V)
    dt= dx**2/(2*(K+0.5*dx*abs(V)) + abs(V)*dx + Fmax*dx**2)
    NT = int(Time/dt)

### MAIN PROGRAM

    for j in range(1,NX-1):
        F[j]= np.exp(-1.e10*(x[j]-0.5)**8)*Fmax
        T[j] = Tref
    T[0]= Tref

### MAIN LOOP
    for n in range(0,NT):

        for j in range(1,NX-1):

            Diffusion = (K+0.5*dx*abs(V))*(T[j-1]-2*T[j] + T[j+1])/(dx**2)
            Advection = V*(T[j+1]-T[j-1])/(2*dx)
            RHS[j] = dt*(Diffusion - Advection + F[j])
```

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

        T[NX-1] = 2*T[NX-2] - T[NX-3]

    j1 = max(0,np.amax(T-Tcible))
    j2 = 0.5*V**2

    if (n == NT-1):
        cost[jctrl]= j1 + j2
        cost1[jctrl]= j1
        cost2[jctrl]= j2
        cvtest = np.linalg.norm(RHS)
        print('CV : ', cvtest, ' Coût = ',cost[jctrl])
        plotlabel = "CNTRL = %1.2f" %(ctrlsave[jctrl])
        plt.plot(x,T,label=plotlabel)


plt.title("Solution")
plt.xlabel('x/L')
plt.ylabel('T/Tref')
plt.grid()
```

```
V= 0.0
CV :  0.3196300243304936  Coût =  2.1249886662321553
V= 0.1
CV :  0.1602959414670268  Coût =  1.758339604821734
V= 0.2
CV :  0.10514374894753067  Coût =  1.0685612517694687
V= 0.30000000000000004
CV :  0.07734603786534618  Coût =  0.6033611158438649
V= 0.4
CV :  0.060034167427764434  Coût =  0.3743624779585563
V= 0.5
CV :  0.03530696743814465  Coût =  0.2604955197479424
V= 0.6
CV :  0.00842644653701746  Coût =  0.20957963051038525
V= 0.7
CV :  0.0007033351064085216  Coût =  0.24499999999999997
V= 0.7999999999999999
CV :  2.3055738234964935e-05  Coût =  0.31999999999999995
V= 0.8999999999999999
CV :  4.1227268945476943e-07  Coût =  0.4049999999999999
V= 0.9999999999999999
CV :  3.257862931498613e-09  Coût =  0.4999999999999999
V= 1.0999999999999999
CV :  1.4686949771010557e-11  Coût =  0.6049999999999999
```

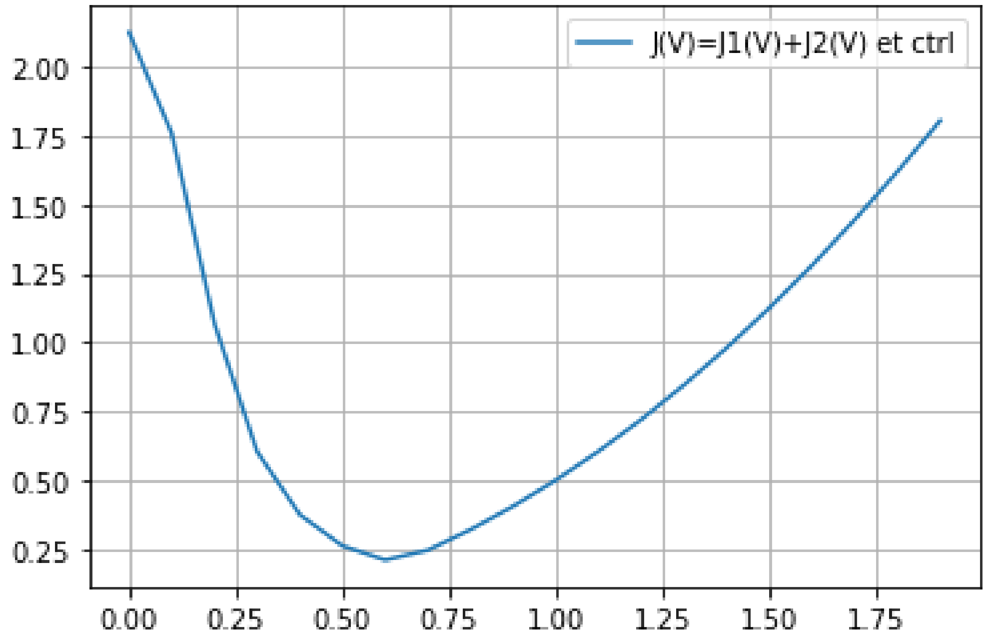
```
V= 1.2
CV :  4.0716085954400287e-14  Coût =  0.72
V= 1.3
CV :  7.302211679180355e-16  Coût =  0.8450000000000001
V= 1.4000000000000001
CV :  7.295722975184307e-16  Coût =  0.9800000000000002
V= 1.5000000000000002
CV :  7.417470075798335e-16  Coût =  1.1250000000000004
V= 1.6000000000000003
CV :  7.39493683972066e-16  Coût =  1.2800000000000005
V= 1.7000000000000004
CV :  7.402545072970038e-16  Coût =  1.4450000000000007
V= 1.8000000000000005
CV :  7.413910030997843e-16  Coût =  1.6200000000000008
V= 1.9000000000000006
CV :  7.380056167885457e-16  Coût =  1.805000000000001
```



La fonction $J(V)$ qui exprime le coût :

[12]:
```
plt.figure()
plt.plot(ctrlsave,cost,label = 'J(V)=J1(V)+J2(V) et ctrl')
plt.grid()
plt.legend()
```

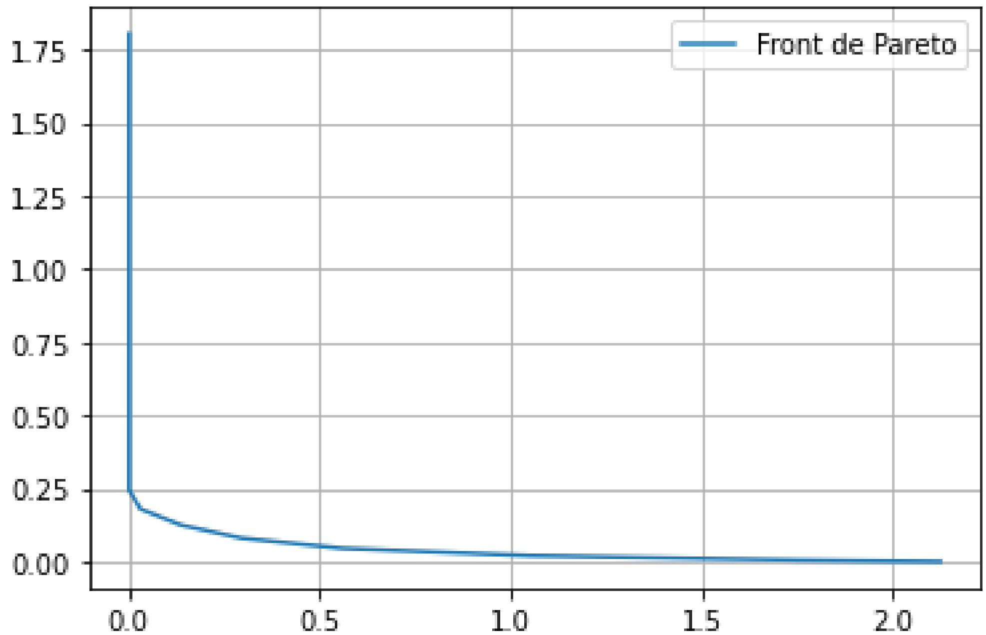
[12]: <matplotlib.legend.Legend at 0x1c4efa700a0>



Le front de Pareto : qui nous montre que $minJ(V) = J(0.6)$

[15]:
```
plt.figure()
plt.plot(cost1,cost2,label='Front de Pareto')
plt.grid()
plt.legend()
```

[15]: <matplotlib.legend.Legend at 0x1c4ef98e8b0>



$V = 0.6$ est donc l'optimum de Pareto, le coût J reste assez faible pour cette valeur V.