Projet 3, Modélisation d'un échangeur thérmique

Zouiche Omar et David Czarnecki

14 Févrirer 2021

1 Modélisation d'un échangeur thermique et dimensionnement d'une isolation

1.1 Problème

On s'intersse à l'isolation thérmique d'une surface plane (un mur, une porte par exemple) d'épaisseur L. On veut trouver le bon niveau d'isolation, ce qui revient à réduire le coéfficient de diffusion thermique afin de diviser par deux le gradient de thermique à la position x = 0.

1.2 Données

Les outils pour modéliser ce phénomène : - On sait d'après la loi Fourrier que l'échange thérmique est proportionnel au gradient de la termpérature T (en kelvin):

$$\overrightarrow{q} = -k \, . \overrightarrow{\text{grad}} \, T$$

Avec : - k la conductivité thérmique (en watts par mètre kelvin) - $\overrightarrow{q}$ la densité du flux thérmique ou du flux de chaleur (en watts par mètres carrées)

- De plus on a le code heat1d.py
- On se limite à une modélisation jusqu'à $t = 1$ (i.e $t \in [0, 1]$)
- La longueur est la surface est $L = 1$

1.3 Contexte

On adapte l'équation de la chaleur à notre problème. On est sur une surface de longueur L, on a une temprérature ambiante de 1°C sur toute la barre sauf à l'extrémité droite où la température est de 4°C, ce qui nous donne une condition de Dirichlet, le côté gauche de la barre quant à lui est isolé, ce qui se traduit par une condition de Neumann homogène. Autrement dit :

$$\begin{cases} \frac{\partial T}{\partial t}(x,t) = k \frac{\partial^2 T}{\partial x^2}(x,t) \\ T(x = L, t) = 4 \\ \frac{\partial T}{\partial x}(x = 0, t) = 0 \end{cases}$$

1.4 Modélisation du problème

Sans isolation On applique une source de chaleur 4 fois supérieure à la température sur le reste de la surface sans isolation, puis on observe les changements de température sur chaque point de

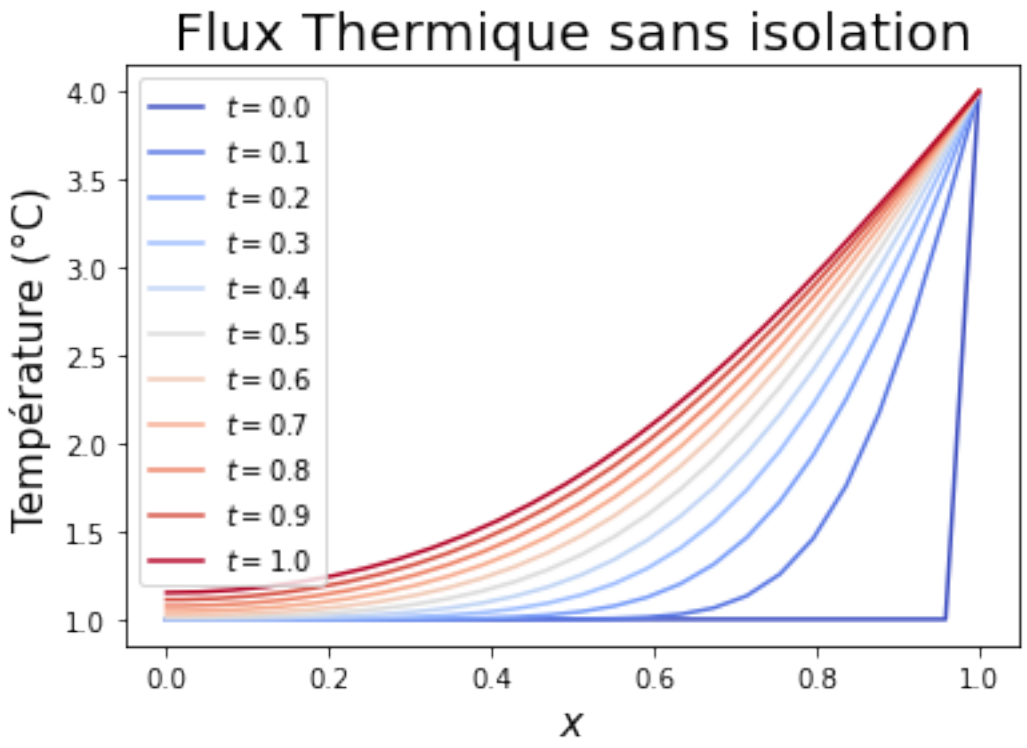
la barre en fonction du temps. Pour cela on adapte le cope heat1d.py à notre situation sachant qu'il faut tenir compte des conditions de Neumann homogène et celle de Dirichlet.

[3]:
```
import numpy as np
import matplotlib.pyplot as plt
import math


K = 0.1              # coefficient de diffusivité thermique
L = 1

NX = 50
X = np.linspace(0,L,NX)
dx = L/(NX-1)
rapport_temps=4

TIME = 1
dt = dx**2/(2*K) # CFL stabilité
NT = int(TIME/dt)

nb_courbes = 10
frame_rate = NT/nb_courbes

T = np.ones((NX)) # on met des 1 car T[0]=T0=1
RHS = np.zeros((NX))
T[NX-1]=rapport_temps

for n in range(0,NT):

  RHS[0]=2*K*dt*(T[1]-T[0])/(dx**2)

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

    RHS[i]=dt*(K*(T[i+1]+T[i-1]-2*T[i])/(dx**2))

  for i in range(0,NX-1):

    T[i] = T[i] + RHS[i]

  if(n%frame_rate == 0):
        plt.plot(X,T,label = r'$t = $'+"%.1f"%(n*dt),
                 color = plt.get_cmap('coolwarm')(float(n)/NT))

print('T[0] sans isolation=',T[0])

plt.plot(X,T,label = r'$t = $'+"%.1f"%(n*dt),color = plt.get_cmap('coolwarm')(n/
 ↪NT))
```

```
plt.title('Flux Thermique sans isolation', fontsize=20)
plt.xlabel('$x$', fontsize =15)
plt.ylabel('Température (°C)',rotation=90,fontsize=15)
plt.legend()
```

```
sans isolation T[0]= 1.1512899807693455
```

[3]: <matplotlib.legend.Legend at 0x1e0c8e65d90>

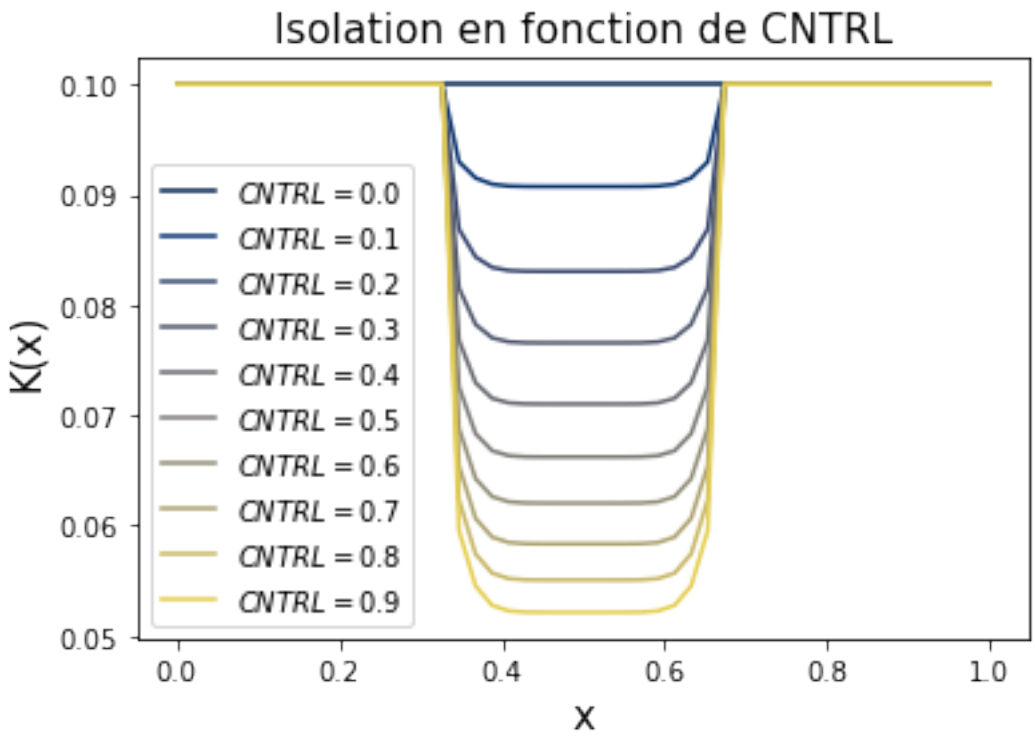


On remarque une augmentation de la température plus importante à droite de la surface. Le changement thérmique arrive même à l'extrémité gauche où la augmente d'environ 0.15°C (1.15°C à $x = 0$).

Avec isolation L'objectif est d'appliquer un isolant afin de réduire le changement de température à l'extrémité gauche de la surface. On applique un isolant de longueur $\frac{L}{3}$ au milieu de la surface qui va baisser la valeur de coefficient thermique k, ce qui fera varier (qu'on nommera $K(x)$) en fonction de CNTRL.

[5]:
```
Rapport_temps=4 #ratio temp entre x=0 et L,
CNTRL = np.linspace(0,1,50)


K = 0.1
L = 1.0
NX = 50

Time = 1
dt=dx**2/(2*K) #CFL stabilité
NT = int(Time/dt)
T0 = 1
KX=np.ones((NX))*K

T = np.ones((NX)) # on met des 1 car T[0]=T0=1
T[NX-1]=rapport_temps

for i in range(0,len(CNTRL)):
   for j in range (1, NX-1):
     if (X[j]<2/3 and X[j]>1/3):
              KX[j] = K /(1+CNTRL[i]*np.exp(-1.e6*(X[j]-0.5)**8))     #modélise␣
 ↪l'isolation centrée de longueur 1/3

   if (i%5 == 0):
     plt.plot(X,KX,label = r'$CNTRL = $'+"%.1f"%CNTRL[i],
              color = plt.get_cmap('cividis')(i/50))


plt.xlabel('x', fontsize=15)
plt.ylabel("K(x)", fontsize=15)
plt.title("Isolation en fonction de CNTRL",
          fontsize = 15)
plt.legend()
```

[5]:
```
<matplotlib.legend.Legend at 0x1e0cb34efd0>
```



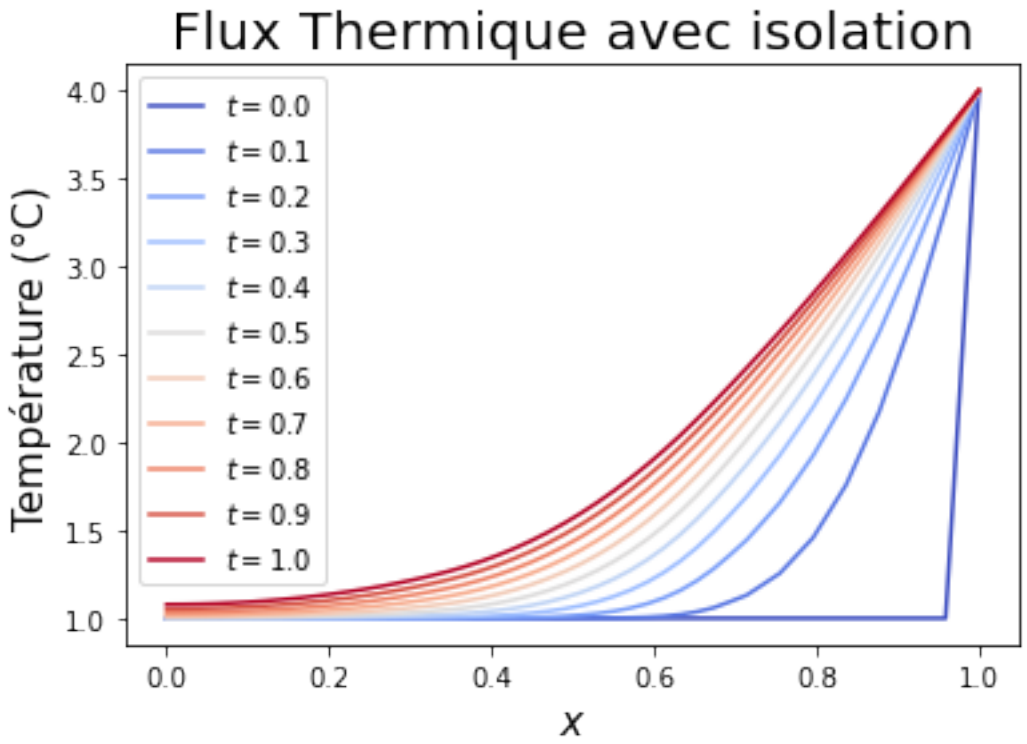
Nous allons voir quel changement cet isolant va apporter à notre surface :

[6]:
```
K = 0.1
L = 1
CNTRL=0.8

NX = 50
X = np.linspace(0,L,NX)
dx = L/(NX-1)
rapport_temps=4

TIME = 1
dt = dx**2/(2*K) # CFL stabilité
NT = int(TIME/dt)

T = np.ones((NX)) # on met des 1 car T[0]=T0=1
RHS = np.zeros((NX))
T[NX-1]=rapport_temps

for j in range (1, NX-1):
     if (X[j]<2/3 and X[j]>1/3):
               KX[j] = K /(1+CNTRL*np.exp(-1.e6*(X[j]-0.5)**8))
     else :
```

```
        KX[j]=K

nb_courbes = 10
frame_rate = NT/nb_courbes

for n in range(0,NT):
  RHS[0]=2*K*dt*(T[1]-T[0])/(dx**2)

  for i in range(1,NX-1):
    RHS[i]=dt*KX[i]*(T[i+1]+T[i-1]-2*T[i])/(dx**2)

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

  if (n%frame_rate==0):
    plt.plot(X,T,label = r'$t = $'+"%.1f"%(n*dt),
             color = plt.get_cmap('coolwarm')(float(n)/NT))



print('avec isolation T[0]=',T[0])
plt.plot(X,T,label = r'$t = $'+"%.1f"%(n*dt),
         color = plt.get_cmap('coolwarm')(n/NT))

plt.title('Flux Thermique avec isolation', fontsize=20)
plt.xlabel('$x$', fontsize =15)
plt.ylabel('Température (°C)',rotation=90,fontsize=15)
plt.legend()
```

```
avec isolation T[0]= 1.0782613787639832
```

[6]: <matplotlib.legend.Legend at 0x1e0cb602790>



L'augmentation de la température à l'extrémité gauche est d'environ $0.8C$, ce qui montre que l'isolant est relativement efficace, on remarque aussi une régularisation de la variation de la température sur la moitié droite de la surface. Ce qui nous conduit à chercher à optimiser l'isolant.

1.5 Amélioration de l'isolant

Soit la fonctionnelle définie par :

$$J(CNTRL) = |\nabla T(0, CNTRL) - \frac{\nabla T(0,0)}{2}|$$

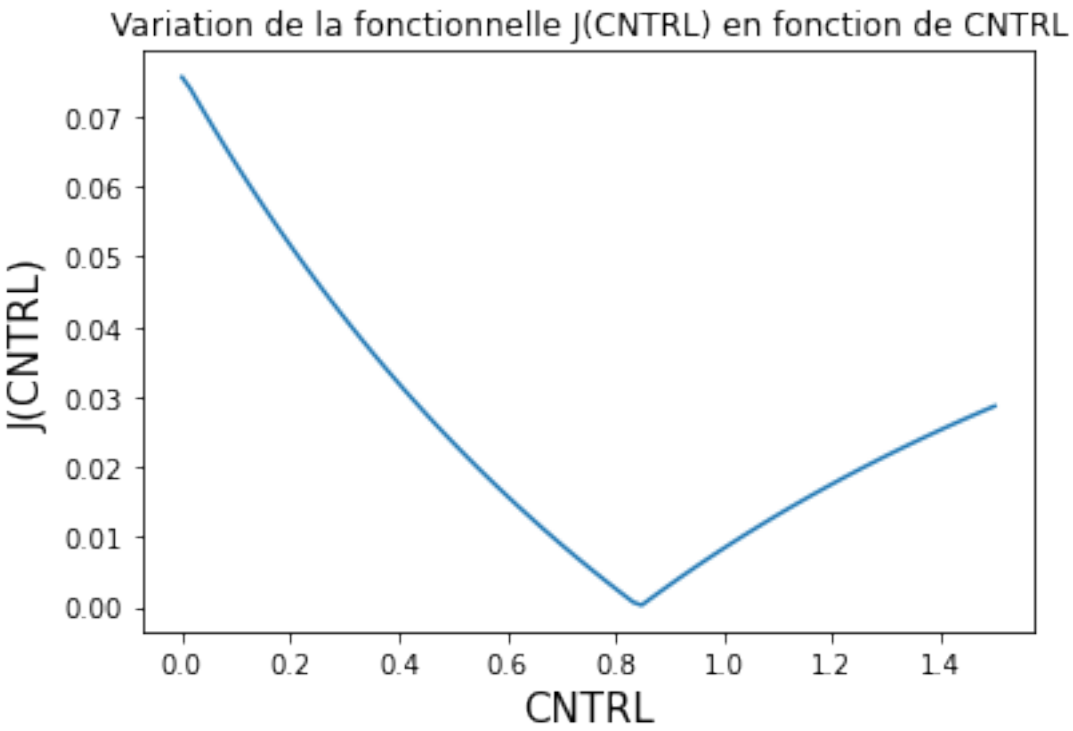
Cette fonctionelle va nous permettre de trouver la valeur de $CNTRL$ afin de réduire le gradient de la température de moitié à l'extrémité gauche. On va donc résoudre l'équation :

$$J(CNTRL) = 0$$

[13]:
```
import numpy as np
import matplotlib.pyplot as plt

K = 0.1
L = 1

NX = 50
X = np.linspace(0,L,NX)
```

```
dx = L/(NX-1)
rapport_temps=4

TIME = 1
dt = dx**2/(2*K) # CFL stabilité
NT = int(TIME/dt)

KX=np.ones((NX))*K

CNTRL=np.linspace(0,1.5,100)
J=np.zeros(len(CNTRL))

for l in range(len(CNTRL)):

  for j in range(0, NX):
      if (X[j]>1/3 and X[j]<2/3):
              KX[j] = K /(1+CNTRL[l]*np.exp(-1.e6*(X[j]-0.5)**8))

  T1 = np.ones((NX)) # on met des 1 car T[0]=T0=1
  RHS = np.zeros((NX))
  T1[NX-1]=rapport_temps

  for n in range(0,NT):
    RHS[0]=2*KX[0]*dt*(T1[1]-T1[0])/(dx**2)

    for i in range(1,NX-1):
       RHS[i]=dt*KX[i]*(T1[i+1]+T1[i-1]-2*T1[i])/(dx**2)

    T1 += RHS

  J[l] = T1[0] - 1
  if l == 0:
     J0 = J[l]
  J[l] = abs(J[l] - 0.5*J0)           #notre fonctionnelle

plt.plot(CNTRL,J)
plt.xlabel('CNTRL', fontsize = 15)
plt.ylabel('J(CNTRL)',fontsize =15)
plt.title('Variation de la fonctionnelle J(CNTRL) en fonction de CNTRL')
```

[13]: Text(0.5, 1.0, 'Variation de la fonctionnelle J(CNTRL) en fonction de CNTRL')



On voit bien que $J(CNTRL)$ admet une borne inférieure sur $]0.8, 0.87[$