TP_3_2_Resolution_numerique_equation_de_diffusion

October 21, 2021

Nous allons résoudre numériquement l'équation différentielle décrivant la diffusion thermique au sein d'une: "Ailette de refroidissement" pour apporter une solution numérique en dehors du régime permanent.

1 Ailette de refroidissement

On considère une ailette d'épaisseur e, de longueur L. En $x = 0$, l'ailette est en contact avec un milieu à la température T_0. Tout le reste de la tige est en contact avec l'air ambiant de température uniforme T_e. On appelle λ la conductivité thermique de l'ailette et h le coefficient de transfert conducto-convectif entre l'ailette et l'air. On pose $\delta = \sqrt{\frac{\lambda e}{2h}}$

On rappelle la loi de Newton: $\delta Q = h(T_s - T_f)dSdt$ à l'interface solide/fluide avec T_s la température du solide et T_f la température du fluide.

2 Solution numérique en régime transitoire

L'équation différentielle vérifiée par $T(x,t)$ s'exprime comme $\frac{\partial T}{\partial t} = D\left[\frac{\partial^2 T}{\partial x^2} - \frac{1}{\delta^2}(T - T_0)\right]$

2.1 Adimensionnement de l'équation aux dérivées partielles

Afin de résoudre cette équation différentielle de manière numérique on va introduire des grandeurs sans dimension qui seront calculées numériquement. Nous allons donc effectuer des changements de variables pour adimensionner cette équation.

- On remplace dans l'équation précédente la température T par $\tilde{T}$ tel que $T = T_e\tilde{T}$.
- On remplace dans l'équation précédente la position x par $\tilde{x}$ tel que $x = \delta \times \tilde{x}$.
- On remplace dans l'équation précédente le temps t par $\tilde{t}$ tel que $t = \frac{\delta^2}{D}\tilde{t}$.

On obtient ainsi l'équation différentielle adimensionnée $\frac{\partial \tilde{T}}{\partial \tilde{t}} = \left[\frac{\partial^2 \tilde{T}}{\partial \tilde{x}^2} - \left(\tilde{T} - 1\right)\right]$

Ces grandeurs sans dimension son divisés par l'échelle caractéristique correspondant à leur évolution. Leur solution numérique sera donc toujours de l'ordre de quelques unitées.

2.2 Conditions initiales et conditions aux limites

Afin de résoudre une équation aux dérivées partielles on a besoin d'y ajouter des conditions initiales et des conditions aux limites.

Initialement l'ailette vient d'être aposée sur la partie chaude. La partie en contact avec la partie chaude atteint instantanément la température T_0. Le reste est encore la température extérieure T_e.

• Ecrire la condition initiale pour $\tilde{T}(\tilde{x} > 0, t = 0)$
• Ecrire la condition au limite pour $\tilde{T}(\tilde{x} = 0, t)$
• Ecrire la condition au limite pour $\tilde{T}(\tilde{x} = \tilde{x}_{max}, t)$ à partir de la condition au limite sur la densité de flux thermique et la loi de Fourier.

2.3 Programmation

Nous allons écrire le programme qui calcule numériquement toutes les valeurs de $T(x,t)$.

Il faut importer les librairies :

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

Il faut ensuite définir les paramètres du problème en début de programme

[ ]:
```
T_0 = 2
T_e = 1
t_max = 1
x_max = 1
```

Il faut ensuite choisir le nombre de points que l'on veut allouer au calcul.

[ ]:
```
n_x = 30
```

Ces variables déterminent les autres.

[ ]:
```
dx = # Calculer le pas entre deux positions à partir de n_x et x_max

n_t = np.int(10*(2*dx**(-2)*t_max)) #  Pour la stabilité de la résolution␣
 ↪numérique nous fixons ce nombre de point temporels

dt = # Calculer le pas entre deux instants à parrtir de n_t et t_max

T = np.zeros((n_x,n_t)) # Initialisation du tableau de zéro dans lequel nous␣
 ↪calculerons les valeurs de température. Le premier indice définira la␣
 ↪position et le second le temps.

T[0,:] = # Remplir dans ce tableau les valeurs limites de température à partir␣
 ↪de $\tilde{T}(\tilde{x} = 0, t)$ et T_0

T[1:,0] = # Remplir dans ce tableau les valeurs initiales de température à␣
 ↪partir de $\tilde{T}(\tilde{x}>0, t=0)$ et T_e
```

• L'équation aux dérivées partielles adimensionnées $\frac{\partial \tilde{T}}{\partial \tilde{t}} = \left[\frac{\partial^2 \tilde{T}}{\partial \tilde{x}^2} - (\tilde{T} - 1)\right]$ peut-être discrétisé à l'aide des formules:

$\tilde{T}(\tilde{x}, \tilde{t} + d\tilde{t}) - \tilde{T}(\tilde{x}, \tilde{t}) = \frac{\partial \tilde{T}}{\partial \tilde{t}} d\tilde{t}$

$\tilde{T}(\tilde{x} + d\tilde{x}, \tilde{t}) - \tilde{T}(\tilde{x}, \tilde{t}) = \frac{\partial \tilde{T}}{\partial \tilde{x}} d\tilde{x}$

$\tilde{T}(\tilde{x} + d\tilde{x}, \tilde{t}) + \tilde{T}(\tilde{x} - d\tilde{x}, \tilde{t}) - 2\tilde{T}(\tilde{x}, \tilde{t}) = \frac{\partial^2 \tilde{T}}{\partial \tilde{x}^2} d\tilde{x}^2$

En déduire une expression de $\tilde{T}(\tilde{x}, \tilde{t} + d\tilde{t})$ en fonction de $\tilde{T}(\tilde{x}, \tilde{t})$, $\tilde{T}(\tilde{x} + d\tilde{x}, \tilde{t})$, $\tilde{T}(\tilde{x} - d\tilde{x}, \tilde{t})$.

• implémenter cette expression pour exprimer les indices de $\tilde{x}$ pour calculer $\tilde{T}(\tilde{x}, d\tilde{t})$, avec $\tilde{x}$ allant de $d\tilde{x}$ à $\tilde{x}_{max} - d\tilde{x}$.

• utiliser la deuxième condition au limite pour calculer $\tilde{T}(\tilde{x}_{max}, d\tilde{t})$

• implémenter ces deux dernière étapes dans une boucle sur les indices de $\tilde{t}$ pour calculer $\tilde{T}(\tilde{x},\tilde{t})$

[ ]:
```
indices = np.arange(n_x-2)+1 # on calcule tous les indices de position à la␣
 ↪fois

for i in range(n_t-1) : # on fait une boucle sur les indices de temps
    T[indices,i+1] = # calcul de la température à l'instant d'après
    T[n_x-1,i+1] = # condition aux limites
```

2.4 Représentation des résultats

Les lignes de commandes ci-dessous permettent de tracer quelques évolutions de températures en fonction de la position ou du temps.

[ ]:
```
x = np.linspace(0,x_max-dx,n_x)
t = np.linspace(0,t_max-dt,n_t)

plt.figure()

plt.subplot(121)
plt.plot(x,T[:,0])
plt.plot(x,T[:,np.int(n_t*(x_max**2)/(4*t_max))])
plt.plot(x,T[:,n_t-1])

plt.subplot(122)
plt.plot(t,T[0,:])
plt.plot(t,T[np.int(n_x/2),:])
plt.plot(t,T[n_x-1,:])

plt.show()
```

Les lignes de commandes ci-dessous permettent de représenter le champs de température de l'ailette à différent instant.

[ ]:
```
y = [0, 1]

plt.figure()

#for i in range(5):
plt.subplot(151)
temp = np.zeros((n_x,2))
temp[:,0] = T[:,np.int(0*(n_t-1)/4)]
temp[:,1] = T[:,np.int(0*(n_t-1)/4)]
plt.contourf(y,x,temp,cmap='hot')
plt.clim([T_0, T_c])

plt.subplot(152)
temp = np.zeros((n_x,2))
temp[:,0] = T[:,np.int(1*(n_t-1)/4)]
```

```
temp[:,1] = T[:,np.int(1*(n_t-1)/4)]
plt.contourf(y,x,temp,cmap='hot')
plt.clim([T_0, T_c])

plt.subplot(153)
temp = np.zeros((n_x,2))
temp[:,0] = T[:,np.int(2*(n_t-1)/4)]
temp[:,1] = T[:,np.int(2*(n_t-1)/4)]
plt.contourf(y,x,temp,cmap='hot')
plt.clim([T_0, T_c])

plt.subplot(154)
temp = np.zeros((n_x,2))
temp[:,0] = T[:,np.int(3*(n_t-1)/4)]
temp[:,1] = T[:,np.int(3*(n_t-1)/4)]
plt.contourf(y,x,temp,cmap='hot')
plt.clim([T_0, T_c])

plt.subplot(155)
temp = np.zeros((n_x,2))
temp[:,0] = T[:,np.int(4*(n_t-1)/4)]
temp[:,1] = T[:,np.int(4*(n_t-1)/4)]
plt.contourf(y,x,temp,cmap='hot')
plt.clim([T_0, T_c])

plt.show()
```

Mesurer à l'aide d'une caméra thermique et comparer vos résultats avec vos calculs numériques.