TP_3_2_Resolution_numerique_equation_de_diffusion

November 24, 2020

Nous allons traiter numériquement l'exercice: "Ailette de refroidissement du TD" pour apporter une solution numérique en dehors du régime permanent.

1 Ailette de refroidissement

On considère une tige de cuivre cylindrique de rayon $a = 5$ mm, de longueur L. En $x = 0$, la tige de cuivre est en contact avec un milieu à la température $T_0 = 330$ K. Tout le reste de la tige est en contact avec l'air ambiant de température uniforme $T_e = 300$ K. On appelle $\lambda = 400$ W.m^{-1}.K^{-1} la conductivité thermique du cuivre et $h = 12$ W.m^{-2}.K^{-1} le coefficient de transfert conducto-convectif entre la tige de cuivre et l'air. On pose $\delta = \sqrt{\frac{\lambda a}{2h}}$

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 analytique en régime permanent

1. Faire un schéma
2. On se place en régime permanent et on considère que la longueur de la tige est quasi infinie. En faisant un bilan d'énergie sur un élément de tige entre x et x+dx, déterminer une équation différentielle vérifiée par $T(x)$. Déterminer le profil de température $T(x)$ en tout point de la tige de cuivre.
3. Toujours en régime permanent on remplace la tige précédente par une tige de longueur $L = 20$ cm. Déterminer $T(x)$.
4. Tracer à l'aide de matplotlib le profil de température $T(x)$ obtenu à la question précédente.

3 Solution numérique en régime transitoire

5. On considère une tige de longueur L mais on ne se place plus en régime permanent. Déterminer alors l'équation différentielle vérifiée par $T(x, t)$.

3.1 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.

• Remplacer dans l'équation précédente la température T par $\tilde{T}$ tel que $T = T_e\tilde{T}$.
• Remplacer dans l'équation précédente la position x par $\tilde{x}$ tel que $x = \delta \times \tilde{x}$.
• Remplacer 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 $\frac{\partial \tilde{T}}{\partial \tilde{t}} = \left[\frac{\partial^2 \tilde{T}}{\partial \tilde{x}^2} - \left(\tilde{T} - 1\right)\right]$

3.2 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.

3.3 Programmation

Nous allons écrire le programme qui calcule numériquement toutes les valeurs de $T(x, t)$.
• Importer la librairie numpy avec le préfixe np
• Importer la librairie matplotlib.pyplot avec le préfixe plt
Il faut ensuite définir les paramètres du problème en début de programme
• Définir et affecter une valeur à une variable $\tilde{T}_0$
• Définir et affecter une valeur à une variable $\tilde{x}_{max}$
• Définir et affecter une valeur à une variable $\tilde{t}_{max}$
Il faut ensuite choisir le nombre de points que l'on veut allouer au calcul.
• Définir le nombre de position $\tilde{x}$ avec la variable $n_x = 30$
Ces variables déterminent les autres.
• Calculer $d\tilde{x}$ le pas entre deux positions $\tilde{x}$
• Pour la stabilité de la résolution numérique le nombre de point temporel doit respecter la formule $n_t = 20\frac{t_{max}}{dx^2}$, et n_t doit être un entier. En utilisant la fonction de numpy np.int(), calculer n_t.
• Calculer $d\tilde{t}$ le pas entre deux instants $\tilde{t}$
• En utilisant la fonction de numpy np.zeros initialiser un tableau $\tilde{T}$ de dimension $n_x \times n_t$ rempli de zéro dans lequel nous calculerons les valeurs de $\tilde{T}(\tilde{x}, t)$. Le premier indice définira la position et le second le temps.
• Remplir dans ce tableau les valeurs initiales de $\tilde{T}(\tilde{x}, 0)$.
• Remplir dans ce tableau les valeurs limites de $\tilde{T}(0, \tilde{t})$, l'autre condition aux limites devra être calulé à chaque itération de la résolution numérique.
• L'équation aux dérivées partielles adimensionnées $\frac{\partial \tilde{T}}{\partial \tilde{t}} = \left[\frac{\partial^2 \tilde{T}}{\partial \tilde{x}^2} - \left(\tilde{T} - 1\right)\right]$ 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 dans un boucle sur 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 $\tilde{T}(\tilde{x}, \tilde{t})$

3.4 Représentation des résultats

• à l'aide de la fonction de numpy np.linspace() définissez des listes $\tilde{x}$, $\tilde{t}$, et si vous voulez représenter aussi la largeur de l'ailette $\tilde{y}$

• à l'aide des fonctions de matplotlib plt.figure(), plt.subplot, plt.plot, plt.show, plt.contourf(), plt.clim, représenter des traces de la température à différent instant, ou à différent endroit, ou des images de la plaque à différent instant.

• à l'aide de la fonction de matplotlib quiver(), vous pouvez après l'avoir calculé, représenter le champ vectoriel de vecteur densité de flux thermique.