Compte rendu
Projet de Calcul Scientifique Numérique

Belpois Vincent
Chomette Hugo

Table des matières

1 Présentation du problème

1.1 Le problème

On modélise un thermocouple par une sphère uniforme de rayon a. Cette sphère, initialement à la température T_0, échange convectivement avec le milieu extérieur, à la température T_∞, selon la loi $\varphi = h(\theta)(T - T_\infty)$.

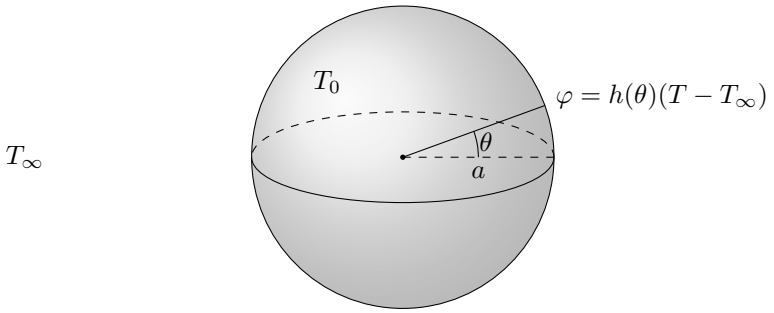

Figure 1: Schéma du Thermocouple

On cherche alors à étudier la réponse de ce thermocouple au cours du temps. Pour cela, on s'impose l'utilisation de la méthode des différences finies

1.2 Interprétation des coéficients

La fonction $h(\theta)$ décrit la dépendance de la convection avec l'angle θ entre la normale à la surface de la sphère et la direction de l'échange thermique. Plus précisément, la quantité de chaleur échangée par convection entre la sphère et le milieu extérieur est proportionnelle à la différence de température entre la sphère et le milieu extérieur, multipliée par $h(\theta)$.

La forme de la fonction $h(\theta) = K(1 + \cos(\theta))$ indique que l'échange de chaleur par convection est maximale lorsque la normale à la surface de la sphère est parallèle à la direction de l'échange thermique et dans le me sens que celui-ci, c'est-à-dire lorsque $\theta = 0°$. Dans ces cas, $h(\theta) = K(1 + 1) = 2K$. L'échange de chaleur par convection est minimale lorsque la normale à la surface de la sphère est dans le sens opposé à celui de l'échange thermique, c'est-à-dire lorsque $\theta = 180°$. Dans ce cas, $h(\theta) = K(1 - 1) = 0$.

La constante K détermine l'intensité de l'échange de chaleur par convection pour un angle θ donné. Plus K est grand, plus l'échange de chaleur par convection est important. K est alors exprimé en $W.m^{-2}.K^{-1}$, tout comme $h(\theta)$.

1.3 Mise en équation du problème

Ce problème revient alors a résoudre l'équation de la chaleur en 3 dimmensions et au cours du temps. De manière générale, l'équation de la chaleur s'écrit de la manière suivante.

$$\rho C_p \left(\frac{\partial T}{\partial t} + u \cdot \nabla T \right) = \nabla \cdot (\lambda \nabla T) + s \tag{1}$$

Dans notre cas, cette équation peut être simplifiée. On a premièrement $u = 0$ car la sphère n'est pas en mouvement. De plus, lambda est constant, on a donc par linéaritée du gradiant $\nabla(\lambda \nabla T) = \lambda \nabla^2 T$. Enfin, le thermocouple n'étant traversé par aucun courant électrique, aucune chaleur n'est produit en sont sein et don cle terme de source s est nul.

L'équation de la chaleur devient alors :

$$\rho C_p \frac{\partial T}{\partial t} = \lambda \nabla^2 T \tag{2}$$

On peut aussi écrire une condition a la surface de la sphère, représentant un échange convectif :

$$\varphi = h(\theta)(T - T_\infty)$$

Avec $h(\theta)$ le coéficient de convection exprimé précédement.

2 Choix du système de coordonnées et du maillage

2.1 Comparaison des systèmes de coordonnées cartésiennes et sphériques

2.1.1 Coordonnées cartésiennes

Le système de coordonnées cartésiennes a l'avantage d'utiliser un repère orthonormé. L'expression des opérateurs tels que le laplacien est donc simple et les équation sont alors simplifiées. Cependant, si le maillage n'est pas régulier selon les trois axes du système de coordonnées cartésiennes, l'expression du pas est alors plus complexe.

AJOUTER PETIT REPERE CARTESIEN

2.1.2 Coordonnées sphériques

Le problème décrivant une sphère, le seul système de coordonnées qui peut être intéressant à utiliser est le système de coordonnées sphériques. Ce système de coordonnées est aussi constitué de trois axes orthonormés:$\overrightarrow{e_r}, \overrightarrow{e_\theta}, \overrightarrow{e_\varphi}$. On utilisera alors pour décrire la position d'un point un triplet de coordonées (r, θ, φ).

AJOUTER PETIT REPERE SPHERIQUE

Pour passer des coordonées cartésiennes aux coordonées sphériques on utilise les équtions suivantes :

$$\begin{cases} r = \sqrt{x^2 + y^2 + z^2} \\ \theta = \arccos\left(\frac{z}{r}\right) \\ \varphi = \arctan\left(\frac{y}{x}\right) \end{cases} \tag{3}$$

De la même manière, pour passer des coordonnées sphériques aux coordonnées cartésiennes on utilise les équations suivantes :

$$\begin{cases} x &= r\sin\theta\cos\varphi \\ y &= r\sin\theta\sin\varphi \\ z &= r\cos\theta \end{cases} \tag{4}$$

2.2 Réécriture des équations adaptée au nouveau système de coordonnées

L'équation de la chaleur Eq.(2) utlise l'opérateur laplacien ($\Delta = \nabla^2$). Cet opérateur s'écrit en coordonnées cartésiennes :

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} \tag{5}$$

En coordonnées sphérique, cet opératreur s'écrit alors :

$$\Delta f = \frac{\partial^2 f}{\partial r^2} + \frac{1}{r}\frac{\partial f}{\partial r} + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2} + \frac{1}{r^2\tan\theta}\frac{\partial f}{\partial \theta} + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial \varphi^2} \tag{6}$$

L'équation de la chaleur Eq.(2) s'écrit alors :

$$\rho C_p \frac{\partial T}{\partial t} = \lambda\left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{1}{r^2\tan\theta}\frac{\partial T}{\partial \theta} + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 T}{\partial \varphi^2}\right) \tag{7}$$

Dans la suite du problème on utilisera la constante $\alpha = \frac{\lambda}{\rho C_p}$ afin de simplifier l'écriture de l'équation ci-dessus.

2.3 Maillage de la sphère

Une sphère peut être mailler de plusieurs mainères dans un espace 3D, certains maillage favoriseront l'uniformité de la taille des mailles comme les maillages quandrangulaires ou en géode par triangulation tandis que d'autres faciliterons la résolution numérique du problème comme le maillage en sphère UV (maillage en faces rectangulaires, formé par des segments assimilables a des lignes de longitude et de latitude sur un globe).

AJOUTER PETIT SPHERE QUANDRAGULAIRE, GEODE(ISO) et UV

On utilisera dans le cadre de notre simulation un maillage UV de la sphère, ce modèle se prettant très bien au systèe de coordonnées sphérique. Cela simplifiera les calculs et l'implémentation des équations.

Les paramètres pour construire ce maillage seront alors `Nr, Ntheta, Nphi`, représentant respectivement le nombre de mailles dans la direction r, la direction θ, et la direction φ

On repèrera par le triplet (i, j, k) les coordonées du noeud du maillage de la sphère.

3 Discrétisation du problème

La méthode utilisée ici est la méthode des différences finies. On choisi d'utiliser l'ordre 1 afin de simplifier les expressions des termes de l'équation de la chaleur, un ordre supérieur pourra être utilisé afin d'augmenter la précision et la stabilité si nécessaire.

3.1 Discrétisation de l'équation de la chaleur

On rapelle l'équation de la chaleur utilisée :

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T \tag{8}$$

Et donc en coordonées sphériques, en prenant en compte l'invariance du champ de température par rapport à φ :

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{1}{r^2 \tan\theta}\frac{\partial T}{\partial \theta} \right) \tag{9}$$

On cherceh a exprimer T^n en fonction uniquement de T^{n-1} On utilise alors un schéma arrière temporel. :

$$\frac{\partial T}{\partial t} \to \frac{T^n - T^{n-1}}{\Delta t} \tag{10}$$

On discrétise ensuite avec un schéma centré les termes issus du laplacien de la température :

$$\frac{\partial^2 T}{\partial r^2} \to \frac{T^{n-1}_{i+1,j,k} - 2T^{n-1}_{i,j,k} + T^{n-1}_{i-1,j,k}}{\Delta r^2} \tag{11}$$

$$\frac{1}{r}\frac{\partial T}{\partial r} \to \frac{1}{r_i}\frac{T^{n-1}_{i+1,j,k} - T^{n-1}_{i-1,j,k}}{\Delta r} \tag{12}$$

$$\frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} \to \frac{1}{r_i^2}\frac{T^{n-1}_{i,j+1,k} - 2T^{n-1}_{i,j,k} + T^{n-1}_{i,j-1,k}}{\Delta \theta^2} \tag{13}$$

$$\frac{1}{r^2 \tan\theta}\frac{\partial T}{\partial \theta} \to \frac{1}{r_i^2 \tan\theta_j}\frac{T^{n-1}_{i,j+1,k} - T^{n-1}_{i,j-1,k}}{2\Delta \theta} \tag{14}$$

On peut alors exprimer T^n :

$$T^n = \alpha \Delta t \left(\frac{T^{n-1}_{i+1,j,k} - 2T^{n-1}_{i,j,k} + T^{n-1}_{i-1,j,k}}{\Delta r^2} + \frac{1}{r_i}\frac{T^{n-1}_{i+1,j,k} - T^{n-1}_{i-1,j,k}}{\Delta r} + \frac{1}{r_i^2}\frac{T^{n-1}_{i,j+1,k} - 2T^{n-1}_{i,j,k} + T^{n-1}_{i,j-1,k}}{\Delta \theta^2} + \frac{1}{r_i^2 \tan\theta_j}\frac{T^{n-1}_{i,j+1,k} - T^{n-1}_{i,j-1,k}}{2\Delta \theta} \right) + T^{n-1} \tag{15}$$

3.2 Discrétisation des condition aux limites

On doit par la suite discrétiser notre condition à la surface de la sphère que l'on rappelle être :

$$\varphi = h(\theta)(T - T_\infty) \tag{16}$$

On a donc par la loi de Fourier :

$$h(\theta)(T(r=a) - T_\infty) = -\lambda \left.\frac{\partial T}{\partial r}\right|_{r=a} \tag{17}$$

On a alors, par les discrétisation des dérivées partielles précédentes :

$$T^n_{i,j,k} = \left(h(\theta) + \frac{\lambda}{\Delta r}\right)^{-1} \cdot \left(h(\theta)T_\infty + \lambda\frac{T_{i-1,j,k}}{\Delta r}\right) \tag{18}$$

Les équations Eq.(15)et Eq.(18) nous permmettrent alors d'exprimer, en tout point de la sphère, la température en fonction des températures de la sphère l'instant précédent

4 Simulation Python

Pour implémenter une telle simulation, nous avons utilisé Python comme language de programmation pour sa facilitée et sa capacitée a visualiser des données à l'aide de bibliothèques simples. La capacitée à executer du code sans compiler nous a aussi permis d'itérerplus rapidement et de débugger simplement le code.

4.1 Structure du code

Le code est composé de quelques fonctions simples et notre situation ne semblait pas demander a introduire de la programation orientée objet, car nous avons décidé de représenter les maillages et les simualations commme de simples tableaux `numpy`.

Ainsi, on commence dabord par fixer nos constantes de la simulation comme les paramètres de maille, les constantes du matériau considéré, les températures (T_0 et T_∞), le facteur de convection, etc...

On construit ensuite le maillage de la sphère.

Enfin on itère la simulation en stockant chaques maillages de la simulaiton dans une liste.

Finalement, on affiche cette simulation et on enregistre une animation de celle-ci.

4.2 Implémentation du maillage

4.2.1 Maillage et liste de maillages

Chaque maillage de a sphère est représenté par une tableau `numpyarray` de dimmension 4. Les 3 premières coordonnées sont utilisées pour la position (soit x, y, z soit r, θ, φ) tandis que la quatrième est utilisée pour représenté la température de ce noeud. Ainsi on créée un objet `maillage` de la manière suivante :

```
    maillage = np.zeros((Nr,Ntheta,Nphi,4))
    maillage[:,:,:,3] = T0                    #set every node temperature to T_0
    for i in range(Nr):
        for j in range(Ntheta):
            for k in range(Nphi):
                maillage[i,j,k,:3] = ((i)*a/(Nr), (j+0.5)*math.pi/Ntheta, 2*k*math.pi/Nphi)
```

Les coordonées sont alors données par Eq.(4). On initialise tous les noeuds à la température T_0, tempréature initiale du thermocouple supposé uniforme.

On introduit aussi une fonction permettant d'obtenir le maillage constitué des coordonnées caertésiennes de chaque noeud afin de par la suite pouvoir visualiser ce mailage :

```
    def polar_to_cartesian(polar):
        cart = np.copy(polar)
        cart[:,:,:,0] = polar[:,:,:,0] * np.sin(polar[:,:,:,1])*np.cos(polar[:,:,:,2])
        cart[:,:,:,1] = polar[:,:,:,0] * np.sin(polar[:,:,:,1])*np.sin(polar[:,:,:,2])
        cart[:,:,:,2] = polar[:,:,:,0] * np.cos(polar[:,:,:,1])
        return cart
```

On construit alors une simulation comme étant une liste de différents maillage (les maillages sont physiquement identiques mais la valeur de la température à chaque noeud est différente). Cela peut se voir dans la fonction `simulation` qui ajoute a la liste de maillage, le nouveau maillage calculé par la fonction `compute_next_step`:

```
    def simulation():
    dt = 0.000001
    meshes = []
    meshes.append(maillage)
    for i in range(25000):
        print(i)
        meshes.append(compute_next_step(meshes[i], dt))
    return meshes
```

`meshes` est alors l'ensemble des maillage de notre simulation.

4.2.2 Visualisation des maillages et des simulations

On a décidé d'utiliser la librairie `matplotlib` afin de visualiser nos données car cette librairie peut à la fois permettre de réaliser des visualisation 3D et de simples graphs 2D.

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.animation

fig = plt.figure()
fig.set_figheight(15)
fig.set_figwidth(15)
ax = fig.add_subplot(111, projection='3d')

ax.axes.set_xlim3d(left=-0.004, right=0.004)
ax.axes.set_ylim3d(bottom=-0.004, top=0.004)
ax.axes.set_zlim3d(bottom=-0.004, top=0.004)

norm = matplotlib.colors.Normalize(vmin=380, vmax=451)
```

On commence par intialiser une figure aec une projection de type '3d'. On fixe alors les limites de l'affichage. Et enfin, on créé la variable `norm` qui sera utilisé lors de l'afichage des températures par une plage de couleurs.

On doit alors, pour afficher une animation des points qui changent de couleur au cours du temps, créer une fonction qui met a jour la visualisation, et change les données envoyés à `matplotlib` :

```
def update_graph(num):
    factor = 100
    global k
    mesh_c = polar_to_cartesian((simulation_meshes[num*factor]))
    graph._offsets3d = mesh_c[:,:,:,0].ravel(), mesh_c[:,:,:,1].ravel(), mesh_c[:,:,:,2]
                                                    .ravel()

    graph.set_facecolor(cmap_inferno(norm(mesh_c[:,:,:,3].ravel())))

    title.set_text('3D Test, num={}'.format(num*factor))
    return title, graph,
```

Le paramètre `factor` est utlisé pour n'afficher que certaines étapes de la simulation, ici toutes les 100 mises a jours du maillage. Dans le cas contraire, la visualisation de l'entièreté de la simulation prendrait des heures.

On peut alors afficher la simulation de la manière suivante :

```
mesh_c = polar_to_cartesian((simulation_meshes[0]))
graph = ax.scatter(mesh_c[:,:,:,0].ravel(), mesh_c[:,:,:,1].ravel(), mesh_c[:,:,:,2].
                                                ravel())
ani = matplotlib.animation.FuncAnimation(fig, update_graph,249, blit=False)

plt.show()
ani.save("simulation.gif", fps=10)
```

Ici, 249 représente la dernière itération de la simulation que l'on souhaite voir (qui sera multiplié par `factor` et donc creprésente ici l'itération 24900). On sauvegarde aussi l'animation créé pour de prochaines visualisations.

4.3 Implémentation des équations

Les équations discutés dans les parties précédentes sont implémentés dans la fonction `compute_next_step`. Cette fonction prend en paramètre le maillage au temps n et le pas temporel `dt` et renvoit le nouveau maillage au temps $n+1$.

Dans l'extrais de code qui suit, certaines parties ont été omises afin d'en facilité la lecture. en effet on remarque que plusieurs cas selon la valeur de `i` ont été disociés. Cela est dû au fait que l'expression du laplacien en différente en 0 et la présence de la condition à la surface de la sphère (en $r = a$). On remarque aussi que des distinctions de cas ont été faites selon la valeur de `j`. Cela provient des problèmes d'indices ; les indices ne forment pas une boucle lorsque que l'on atteint le dernier indice.

```
def compute_next_step(p_mesh, dt):
    Nr, Ntheta, Nphi, _ = np.shape(p_mesh)
    n_mesh = np.copy(p_mesh)

    Delta_r = a / Nr
    Delta_theta = math.pi / Ntheta
    Delta_phi = 2 * math.pi / Nphi
    for i in range(0, Nr):
        if i == 0:
            ...
        elif i == Nr - 1:
            ...
        else:
            r = i * Delta_r
            for j in range(Ntheta):
                theta = (j+1.5) * Delta_theta
                if j == 0:
                    ...

                elif j == Ntheta - 1:
                    ...

                else:
                    for k in range(Nphi):
                        phi = (k+1) * Delta_phi
                        n_mesh[i,j,k,3] = alpha * dt *
                    ((p_mesh[i+1,j,k,3]-2*p_mesh[i,j,k,3]+p_mesh[i-1,j,k,3])/(Delta_r**2)
                    + 1/r*(p_mesh[i+1,j,k,3]-p_mesh[i-1,j,k,3]) / Delta_r
+ 1/r**2 * (p_mesh[i,j+1,k,3] - 2 * p_mesh[i,j,k,3] + p_mesh[i,j-1,k,3]) / (Delta_theta**2)
+ 1/(r**2 * math.tan(theta)) * p_mesh[i,j+1,k,3] - p_mesh[i,j-1,k,3] / (2* Delta_theta))
+ p_mesh[i,j,k,3]

    return n_mesh
```

4.4 Simulations

4.4.1 Choix des valeurs des paramètre

Choix des pas, des dimensions, de Delta t, du temps de simulation

4.4.2 Test de l'influence de certains paramertres

4.4.3 Simulations

5 Conclusion

Conclure sur la méthode utilisé, l'implémentation, la convaincance des résultats.

Finir sur ce que cela nous a apris, la facilitée d'implémenter vs la difficultée à discrétiser le probblème et choisir la valeur des paramètres