Méthode des différences finies : propagation thermique dans une poutre

PSI 2022/2023

En ingénierie numérique, la *méthode des différences finies* est une technique très courante de recherche de solutions numériques d'équations différentielles aux dérivées partielles. Elle est notamment utilisée en mécanique des fluides pour résoudre l'équation de Navier-Stokes. L'objectif de ce TP est de présenter cette méthode sur l'exemple simple de l'équation de la chaleur à une dimension.

I Présentation de la méthode

I.1 Mise en équation du problème

On considère une tige métallique homogène, connectée à ses deux extrémités à deux thermostats de température $T_c = 40°\text{C}$ et $T_f = 20°\text{C}$ et on note $T(x,t)$ le champ des températures dans la tige. Ce champ est solution de l'équation de diffusion de la chaleur ci-dessous :

$$\frac{\partial T}{\partial t}(x,t) = D\frac{\partial^2 T}{\partial x^2}(x,t) \tag{1}$$

Le problème est adimensionné : la tige a une longueur $L = 1$ et on mesure la position spatiale d'un point sur la tige par une absisse $x \in [0,1]$. La présence des thermostats impose alors les conditions aux limites suivantes :

$$\begin{cases} T(x=0,t) & = T_c \\ T(x=L,t) & = T_f \end{cases} \quad \forall t \in \mathbb{R}^+ \tag{CL}$$

On suppose par ailleurs que la barre est initialement à la température T_f sauf en $x = 0$ où la température vaut T_c :

$$T(x > 0, t = 0) = T_f \tag{CI}$$

On cherche à déterminer l'évolution du champ des températures sur l'intervalle temporelle $[0, t_{\max}]$ où $t_{\max}$ représente la durée de la simulation.

I.2 Discrétisation spatiale et temporelle

Pour envisager une résolution approchée, nous allons subdiviser les intervalles de temps $[0, t_{\max}]$ et d'espace [0,1] :

- comme pour la méthode d'Euler classique (appelé schéma d'Euler explicite), l'intervalle de temps est subdivisé en N_t sous-intervalles de même durée `dt`.
- l'intervalle d'espace est subdivisé en N_x sous-intervalles de longueur `dx`.

Le champ $T(x,t)$ est alors modélisé par un tableau *spatio-temporel* de taille $N_x \times N_t$:

$$T(x,t) \approx T(x_i,t_j) \approx \texttt{T[i, j]} \text{ avec } 0 \leq i \leq N_x - 1 \text{ et } 0 \leq j \leq N_t - 1$$

La prise en compte des conditions initiales et des conditions aux limites donne le tableau ci-dessous. Toutes les valeurs inconnues sont fixées à 0.

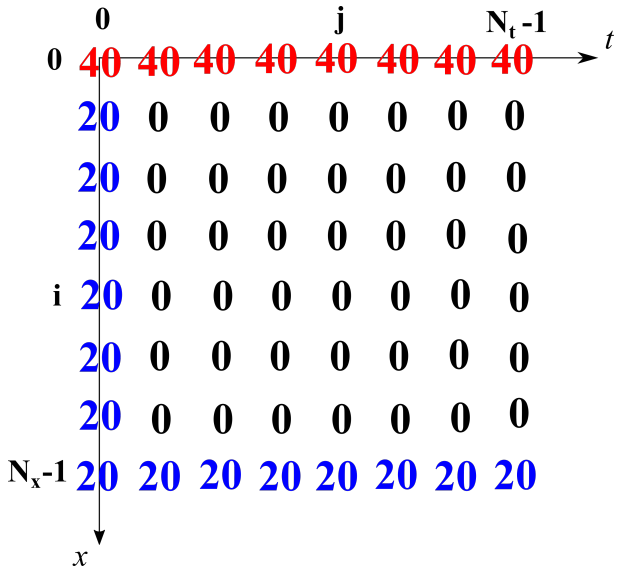


FIGURE 1 – *Initialisation du tableau des températures*

Pour calculer les termes restants, il faut discrétiser l'équation de la chaleur. La discrétisation la plus simple du terme de gauche de l'équation de diffusion s'écrit en utilisant le schéma d'Euler classique :

$$\frac{\partial T}{\partial t}(x,t) \approx \frac{\texttt{T[i, j+1]} - \texttt{T[i, j]}}{\texttt{dt}}$$

La dérivée d'ordre 2 du terme de droite s'approxime en :

$$\frac{\partial^2 T}{\partial x^2}(x,t) = \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}(x,t) - \frac{\partial T}{\partial x}(x-\texttt{dx},t)\right) \approx \frac{1}{\texttt{dx}}\left(\frac{\texttt{T[i+1, j]} - \texttt{T[i, j]}}{\texttt{dx}} - \frac{\texttt{T[i, j]} - \texttt{T[i-1, j]}}{\texttt{dx}}\right)$$

L'équation de diffusion discrétisée s'écrit alors :

$$\boxed{\texttt{T[i, j+1]} = \texttt{T[i, j]} + \texttt{r}\,(\texttt{T[i+1, j]} - 2\texttt{T[i, j]} + \texttt{T[i-1, j]})} \tag{2}$$

avec $\texttt{r} = \texttt{D}\frac{\texttt{dt}}{\texttt{dx}^2}$. Ce schéma est *explicite* dans le sens où le champ des températures à la date t_{j+1} s'exprime directement en fonction du champ des températures à la date t_j.

Le remplissage du tableau s'effectue en appliquant l'affectation (2) pour tous les éléments nuls du tableau (cf figure ci-dessous).

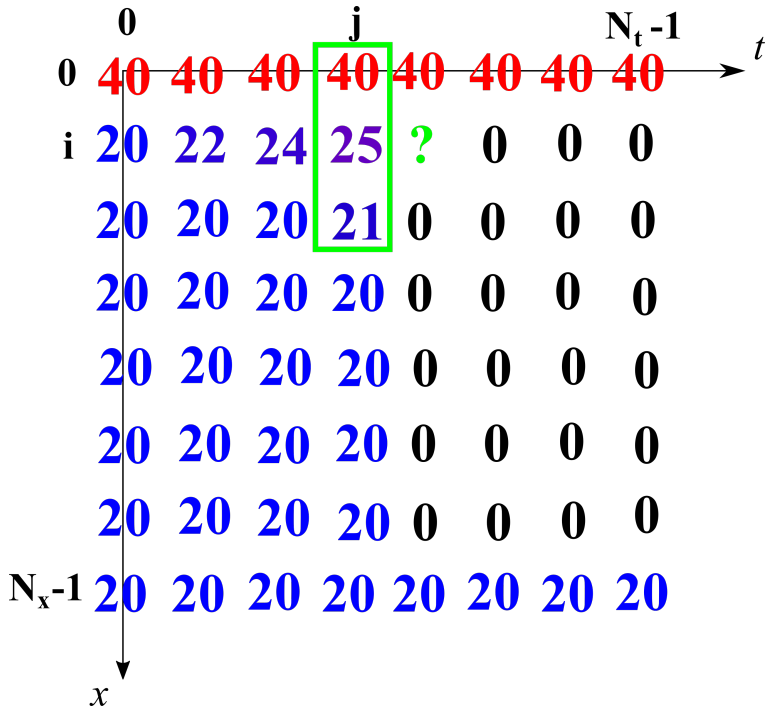


FIGURE 2 – *Remplissage du tableau des températures. Les élements du tableau nécessaire au calcul de* `T[i,j+1]` *sont encadrés en vert.*

II Implémentation

On utilisera la bibliothèque `numpy` dont on rappelle en annexe quelques commandes.

II.1 Schéma d'Euler explicite

Remarque préliminaire : pour remplir les éléments d'un tableau, on pourra utiliser des boucles `for` *mais pour des raisons de vitesse d'exécution, il pourra être préférable de s'en passer en utilisant la méthode du slicing.*

1. Ecrire une fonction `initialisation(Nx,Nt,Tc,Tf)` renvoyant un tableau numpy `T` de taille $N_x \times N_t$ et vérifiant les conditions aux limites et initiales définies par les équations (CL) et (CI).

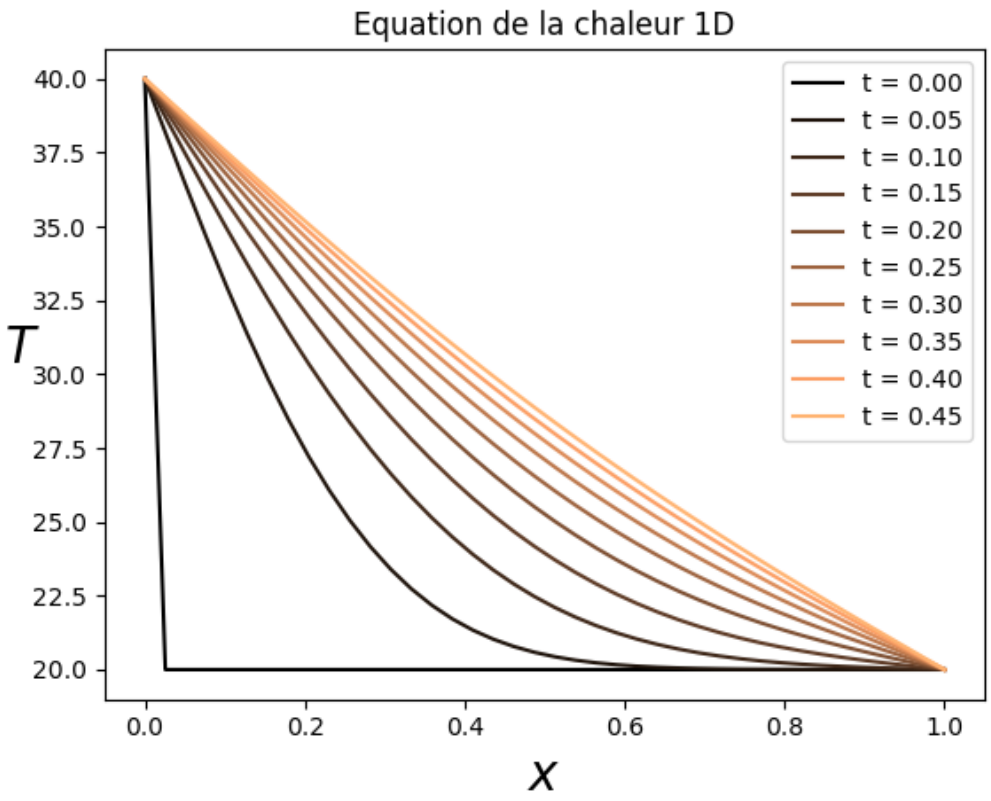
2. Ecrire une fonction `schema1(T,r)` qui remplit toutes les colonnes du tableau `T` précédent à l'aide du schéma donné par l'équation (2).

3. On choisit pour la simulation une durée $t_{\max} = 0.5$. L'intervalle de temps est découpé en $N_t = 1000$ parties. L'intervalle d'espace est découpé en $N_x = 40$ parties. On prendra un coefficient de diffusivité $D = 0.5$. Obtenir le champ des températures vérifiant l'équation de diffusion et compatible avec les (CL) et la (CI). En utilisant la fonction `plot` de la bibliothèque `matplotlib.pyplot`, tracer le profil des températures tous les 100 pas. On pourra utiliser la portion de code suivant :

```
plt.figure(1)
```

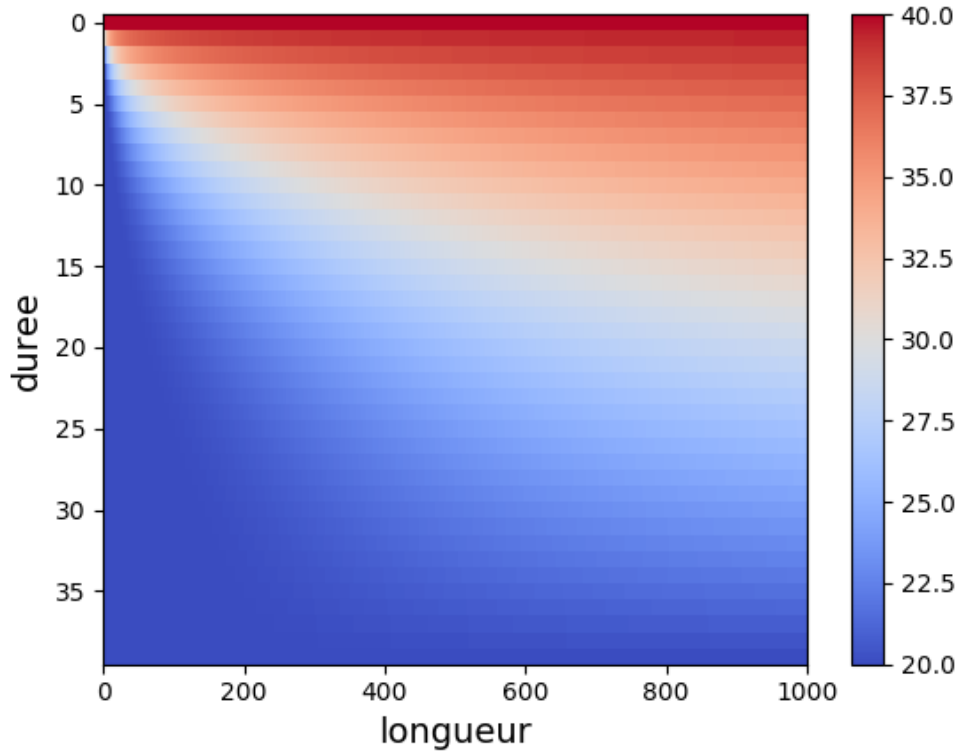
```
plt.cla()
X = np.linspace(0,1,Nx)
for j in range(Nt):
    if j % 100 == 0:
        plotlabel = "t = %1.2f" %(j * dt)
        plt.plot(X,T[:,j], label=plotlabel,color = plt.get_cmap('copper')(
    float(j)/Nt))


plt.xlabel(u'$x$', fontsize=20)
plt.ylabel(u'$T$', fontsize=20, rotation=0)
plt.title(u'Equation de la chaleur 1D')
plt.legend()
plt.show()
```



4. Visualiser l'évolution du champ des températures sur un graphique bidimensionnel à l'aide du code ci-dessous :

```
plt.figure(2)
plt.imshow(T,cmap=plt.cm.coolwarm,alpha=1)
plt.colorbar()
plt.axis('tight')
plt.ylabel(u'duree',fontsize=14)
plt.xlabel('longueur',fontsize=14)
plt.show(t)
```



5. Visualiser l'évolution de la température dans la tige pour $N_x = 45$ et $N_x = 46$. Que se passe-t-il pour ce dernier cas ?

II.2 Schéma d'Euler implicite

On observe que le schéma de discrétisation précédent devient instable lorsque `Nx` devient trop grand. On choisit alors d'utiliser un schéma d'Euler dit *implicite*[1]. Ce schéma consiste à évaluer la dérivée d'ordre 2 dans le terme de droite en faisant la moyenne de l'approximation d'Euler de cette dérivée en x_i et en x_{i+1} :

$$\frac{\partial^2 T}{\partial x^2}(x,t) = \frac{1}{2}\left(\frac{1}{\mathrm{d}x}\left(\frac{\partial T}{\partial x}(x,t) - \frac{\partial T}{\partial x}(x-\mathrm{d}x,t)\right) + \frac{1}{\mathrm{d}x}\left(\frac{\partial T}{\partial x}(x+\mathrm{d}x,t) - \frac{\partial T}{\partial x}(x,t)\right)\right)$$

L'équation discrétisée correspondante s'écrit :

$$\texttt{T[i, j+1]} = \texttt{T[i, j]} + \frac{\texttt{r}}{2}\left(\texttt{T[i+1, j]} - 2\texttt{T[i, j]} + \texttt{T[i-1, j]} + \texttt{T[i+1, j+1]} - 2\texttt{T[i, j+1]} + \texttt{T[i-1, j+1]}\right) \tag{3}$$

Il apparaît alors que le calcul du champ des températures à la date t_{j+1} ne dépend plus seulement des températures à la date antérieure t_j mais fait aussi intervenir la température aux points x_{i-1} et x_{i+1} à la date t_{j+1}.

Pour obtenir le champ des températures, on réécrit l'équation (3) sous la forme matricielle :

$$AT_{j+1} = MT_j \tag{4}$$

1. Il s'agit en réalité d'un cas particulier de schéma implicite appelée schéma de Cranck-Nicolson.

où T_j est un vecteur colonne de taille `Nx` contenant le champ des températures à la date t_j et A et B sont deux matrices tridiagonales de dimension $N_x \times N_x$ telles que :

$$T_j = \begin{pmatrix} T[0,j] \\ \vdots \\ T[i,j] \\ \vdots \\ T[N_x-1,j] \end{pmatrix}, \; A = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -\frac{r}{2} & 1+r & -\frac{r}{2} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -\frac{r}{2} & 1+r & -\frac{r}{2} \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}, \; M = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \frac{r}{2} & 1-r & \frac{r}{2} & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & \frac{r}{2} & 1-r & \frac{r}{2} \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

T_{j+1} est donc la solution d'une équation matricielle de la forme $AX = B$.

6. Ecrire une fonction `tridiagonale(n,c1,c2)` prenant en argument un entier et deux flottants et renvoyant la matrice tridiagonale de taille $n \times n$ suivante :

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ c_2 & c_1 & c_2 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & c_2 & c_1 & c_2 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

7. Obtenir le champ des températures `T[i,j]` à l'aide de l'équation (4).

8. Visualiser l'évolution du champ des températures sur un graphique bidimensionnel comme précédemment. Vérifier la stabilité numérique de la méthode.

Annexe `numpy`

`np.zeros((n,m))`	crée un array $n \times m$ dont les éléments sont nuls.
`np.linspace(a,b,n)`	crée un vecteur de n valeurs régulièrement espacées de a à b.
`linalg.inv(M)`	renvoie l'inverse de M
`linalg.solve(A,B)`	renvoie X tel que $AX = B$
`M.dot(V)`	renvoie le produit matriciel d'une matrice M par un vecteur V
`M1.dot(M2)`	renvoie le produit matriciel d'une matrice $M1$ par une matrice $M2$

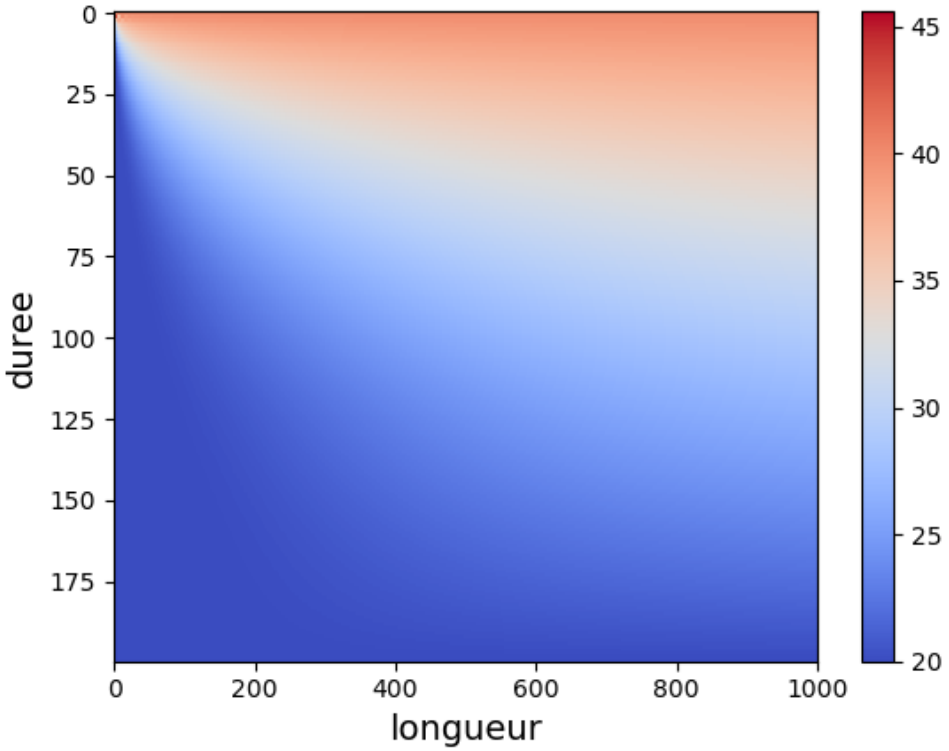


FIGURE 3 – *Evolution de la température dans la barre pour un découpage en 200 morceaux.*