TD D2 : Simulation numérique d'un régime transitoire de conduction thermique monodimensionnelle

Nous ne chercherons pas à adimensionner les équations ni à « simplifier » les valeurs numériques. Les ordres de grandeur doivent être immédiatement interprétables par le sens commun.
Les longueurs seront donc exprimées en m et les durées en secondes. Pour les durées assez élevées des régimes transitoires, on présentera tout de même les dates des profils successifs en minutes en légende de graphe.

Nous nous limiterons au fer comme matériau conducteur thermique. Ses qualités thermiques n'intervenant dans l'équation de la diffusion thermique que par la diffusivité, vous confirmerez puis utiliserez la valeur : $D=2{,}345.10^{-5}\ m^2.s^{-1}$

Les bibliothèques à importer seront **numpy** (renommée np), **matplotlib.pyplot** (renommée plt), **time** pour l'estimation de durées de calcul et **scipy.integrate** pour la résolution numérique d'équations différentielles ordinaires par **odeint.**

I. La barre de Fer entre deux thermostats

I.A. Modélisation de la situation d'un ancien TP de conduction thermique

« Une barre de fer à température initiale uniforme T0=20°C est ***calorifugée latéralement*** *et en* ***contact thermique parfait à ses deux extrémités*** *avec deux thermostats : l'un chaud à gauche constitué d'une* ***eau bouillante à température Tc=100 °C****, l'autre froid à droite maintenu* ***à Tf=0°C grâce à un mélange eau-glace*** *brassé régulièrement. »*

L'équation de diffusion de la chaleur au sein de ce solide sera discrétisée :
- spatialement : on choisira une barre de L=1m de longueur que l'on découpera en Nx=100 tranches de dx=1cm dans un premier temps
- temporellement : on choisira de diviser une durée totale de tau=20000 secondes en Nt=10000 pour un pas temporel de dt=2s

I.B. Programme « Barre de fer 1» : discrétisation de l'équation différentielle

Un profil initial de Nx+2 valeurs de températures sera construit en une ligne avec np.concatenate
Les valeurs des températures de thermostats ne doivent pas varier dans le temps.

L' augmentation de la température de chaque tranche i lors d'un pas temporel sera notée dT[i] (i ème élément d'un tableau dT) .
On exprimera dT[i] en fonction de D,dx,dt, T[i-1],T[i] et T[i+1] à une itération temporelle j quelconque.
On demandera par exemple un affichage de profil de températures toutes les 20 minutes (de la simulation) **(pas tous les profils de température calculés toutes les 2 secondes !)**
On pourra utiliser les « cmap » en associant un dégradé de couleurs à l'écoulement du temps :

plt.plot(x,T,'+',label=plotlabel,color = plt.get_cmap('winter')(float(j/Nt)))

On pourra imprimer ces courbes pour illustrer le cours de thermique par cet exemple d'évolution d'un profil de température lors d'un régime transitoire.
Le programme devra également restituer le temps de calcul de ces Nt profils (pour comparaison avec les variantes suivantes)

I.C. Programme « Barre de fer 2» : une astuce simplifiant les calculs

Montrer, à partir de la version discrétisée de l'équation différentielle, que l'on peut tout simplement proposer de donner à la température d'une tranchette i à l'instant j la valeur moyenne de ses voisines à l'instant j-1. Donner alors la relation de dépendance entre le pas spatial dx et le pas temporel dt pour que cette proposition reste compatible avec l'équation de la diffusion de chaleur.

Modifier alors le programme précédent en exprimant la nouvelle expression de dT[i]
Quel gain en temps d'exécution ?

I.D. Et si on utilisait les opérations sur les tableaux (« Barre de fer 3 ») ?

I.D.1. Les possibilités de calcul « vectoriel » de numpy

On peut facilement tirer profit des instructions vectorielles de numpy pour éviter des boucles de programme. Ces « boucles » sont alors gérées par du code compilé et non du code interprété. La bibliothèque numpy est en effet déjà « compilée ».

> *La différence centrale entre compilé et interprété est comme suit : là où* ***le compilateur traduit une bonne fois pour toute un code source en un fichier indépendant exécutable*** *(donc utilisant du code machine ou du code d'assemblage),* ***l'interprète est nécessaire à chaque lancement du programme interprété****, pour traduire au fur et à mesure le code source en code machine. Cette traduction à la volée est la cause première de la lenteur des langages dits interprétés.*

Rappelons donc quelques opérations d'indexation sur les tableaux :

```
>>> tableau = np.array([1,2,3,4,5,6])
>>> tableau[1:4]   # de l'indice 1 à l'indice 4 exclu !!!ATTENTION!!!
array([2, 3, 4])
>>> tableau[:4]    # du début à l'indice 4 exclu
array([1, 2, 3, 4])
>>> tableau[4:]    # de l'indice 4 inclus à la fin
array([5, 6])
>>> tableau[:-1]   # excluant le dernier élément
array([1, 2, 3, 4, 5])
>>> tableau[1:-1]  # excluant le premier et le dernier
array([2, 3, 4, 5])
```

I.D.2. Application à l'obtention du tableau des augmentations de températures

Que représentera dT[1:-1] dans notre programme ?

Combien d'éléments contient ce tableau ? et T[1:-1] ?

Comment écrire la liste des Nx termes éliminant la température du premier thermostat et de la première tranche ?

Et la liste éliminant la température du dernier thermostat et de la tranche le précédant ?

Après avoir vérifié que ces trois tableaux (monodimensionnels) avaient la même dimension, imaginez ces trois tableaux les uns sous les autres (les indices ont été décalés par les appels des « coupes » précédentes). Quelle simple opération de combinaison linéaire des tableaux précédents calculera immédiatement le tableau dT[1:-1] des variations de température de toutes les tranches à chaque itération temporelle ?

Copiez-collez votre programme précédent dans un nouveau fichier python et remplacez la double boucle de calcul des T[i] par l'opération précédente. Nouveau temps d'exécution ?
Conclusion.

II. Pour les codeurs qui ont fini les parties précédentes : simulation du transitoire de la barre Didalab.

On devra modifier les conditions aux limites spatiales qui deviennent :

- une condition de type Neumann (valeur du flux donnée en 0 : 13 W sur 2 cm² fixant la valeur de la dérivée de la température en 0)
- une condition hybride (type Robin ?) reliant la température à sa dérivée locale par une loi conducto-convective de Newton en x=L

Annexe : Barre de Fer entre 2 thermostats : approche discrète