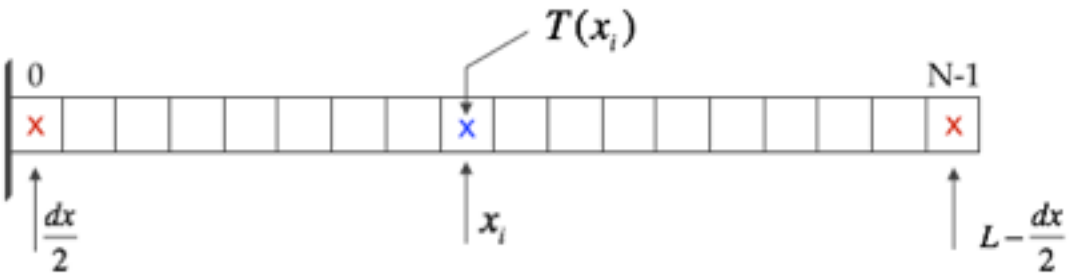
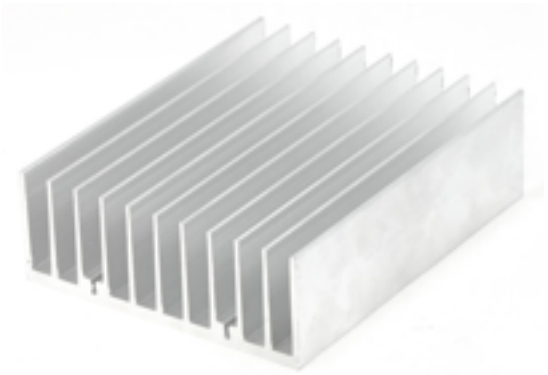
Diffusion thermique dans une ailette

On considère une ailette de refroidissement de processeur en aluminium, de longueur L, de section S, de périmètre P et découpée en N cases élémentaires [cf document de cours]. Ce sujet est inspiré de l'informatique pour tous.



<u>On donne les paramètres suivants pour le code python :</u>



##CONSTANTES
P = 0.062#m périmètre de l'ailette
S = 3.e-5#m2 surface de section
L = 0.02#m longueur de l'ailette

RHO = 8.0e3 #kg/m3 masse volumique
C = 444 #J/kg/K capacité thermique
LAMBDA = 80. #W/m/K conductivité thermique
HNEWTON = 100. #W/m2/K coefficient d'échange
Qent = 333333 #W/m2 flux energie entrant (puissance surfacique)

Tamb = 20#+273.159# en Celcius

I Méthode d'Euler explicite

Pour cette partie on pourra utiliser des listes ou des tableaux numpy.

a - Définir la taille dx des cellules et le pas de temps dt en s'appuyant sur la formule du critère de convergence. On définira aussi la DUREE = 60#s, le nombre de cellule N=20 et on initialisera le temps à t = 0. #s ainsi que l'indice de pas de temps associé n = 0.

b - Construire une liste X de taille N contenant les abscisses des N cellules séparées d'une distance L/N
Construire une liste T contenant les températures de ces cellules initialisées à la valeur Tamb = 20°C.
Construire enfin une liste dT de taille N initialisée avec des 0.0 au sens des flottants.

c - Ecrire une fonction **d2T(Tloc)** :
- qui prend en argument Tloc = [T[i-1], T[i], T[i+1]]
  soit la liste des températures autour de la cellule i dès lors que 0 < i < N-1
- qui retourne la différentielle seconde de la température d^2T/dx^2 au centre de la cellule i.

Cette fonction ne sera pas valide si i = 0 ou i = N-1. Est-ce un problème ?

d - Ecrire une boucle while qui réalise l'intégration d'Euler.
A chaque itération, on calculera la variation de température dT[i] pour 0 < i < N-1 puis dT[0] et dT[N-1] à l'aide des formules de discrétisation vues en classe. On mettra à jour la liste des températures T.

e - Tracer le profil de température obtenu en fin d'intégration à l'aide de la commande plt.plot du module pylpot de matplotlib.

f - **Synthèse** : construire une fonction **tempEvolution** qui réalise l'ensemble du programme précédent, et qui renvoie une liste mesCourbes = [X, T_0, T_1 , ….Tm] contenant la liste des abscisses X dans mesCourbes[0] et ensuite les listes des profils Tk pour 0 <= k < m où m est le nombre de courbes de températures à tracer.
Rq : Tk est donc aussi une liste (profil de température le long de l'aile).

Le nombre m sera passé en argument ainsi que tous les paramètres qui vous semblent intéressants.

exemple : **tempEvolution**(N, DUREE, **m=10**, Tamb=20, P=0.062, S=3.e-5, L=0.02, RHO=8.0e3, C=444, LAMBDA=80., HNEWTON=100., Qent=333333)

Pour aller plus loin :
On peut remplacer le calcul explicite de dT par un simple produit matriciel (commande numpy .dot()).
Il faut pour cela utiliser la matrice donnée en cours et que nous allons aussi utiliser dans la partie qui vient.

II Recherche de la solution stationnaire par résolution d'un système linéaire.

Pour cette partie on devra utiliser des tableaux numpy, ainsi que le module linalg de numpy.
On doit résoudre le système linéaire MT + C = 0

a - Définir la taille dx des cellules à l'aide du nombre de cellules N et de la longueur L.
Construire le tableau numpy X des positions des cellules, à l'aide de la commande linspace (cf cours).
Construire une matrice numpy « vide » M de taille NxN, à l'aide de la commande zeros (cf cours).

b - Remplir la matrice M à l'aide des formules données en cours. On peut ne faire qu'une simple boucle sur la diagonale, et compléter astucieusement autour. On finalise ensuite en M[0, 0] et M[N-1, N-1]

c - Construire de même le tableau numpy associé au vecteur colonne du système B = -C (attention au signe).
d - Résoudre le système à l'aide de la commande linsolve (cf cours).
e - Tracer la solution obtenue à l'aide de la commande plt.plot du module pylpot de matplotlib.

f - **Synthèse :** tout comme à la partie I, écrire une fonction solStationnaire qui renvoie la liste [X, T] conforme au format mesCourbes avec une seule courbe.

Fin