CONCOURS COMMUNS POLYTECHNIQUES

EPREUVE SPECIFIQUE - FILIERE TPC

MODELISATION

Durée : 4 heures

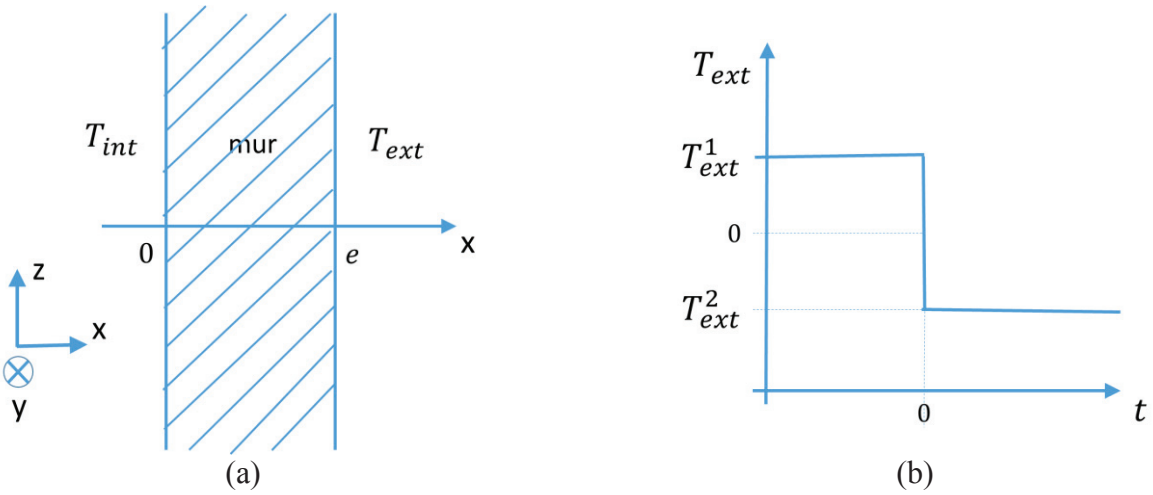
N.B. : le candidat attachera la plus grande importance à la clarté, à la précision et à la concision de la rédaction. Si un candidat est amené à repérer ce qui peut lui sembler être une erreur d'énoncé, il le signalera sur sa copie et devra poursuivre sa composition en expliquant les raisons des initiatives qu'il a été amené à prendre.

Les calculatrices sont interdites

Le sujet comporte deux parties indépendantes. Le candidat précisera au début de sa copie le langage de programmation (Python ou Scilab) qu'il a choisi et toutes les questions seront traitées dans le même langage. Un bonus sera accordé aux copies soignées avec des programmes bien commentés. Plusieurs fonctions du langage Scilab sont rappelées en annexe A. Les candidats choisissant le langage Python pourront utiliser les bibliothèques numpy et matplotlib.pyplot. Une documentation simplifiée de plusieurs fonctions de ces bibliothèques est présente en annexes B et C.

SIMULATION NUMERIQUE DU TRANSFERT THERMIQUE DANS UN MUR EN REGIME TRANSITOIRE

On étudie les transferts thermiques dans le mur d'une maison, figure 1(a). La température à l'intérieur de la maison est constante dans le temps et égale à $T_{int} = 20$ °C. Aux temps négatifs ($t < 0$), la température extérieure est égale à $T_{ext1} = 10$ °C. A $t = 0$, elle chute brusquement à $T_{ext2} = -10$ °C et elle reste égale à cette valeur aux temps positifs ($t > 0$), figure 1(b). On souhaite étudier l'évolution du profil de température dans le mur au cours du temps.



(a) (b)

Figures 1 (a) - Schéma du mur étudié. (b) - Evolution de la température extérieure au cours du temps.

Le mur a une épaisseur $e = 40$ cm. Les propriétés physiques du mur sont constantes : conductivité thermique $\lambda = 1{,}65\ \mathrm{W.m^{-1}.K^{-1}}$, capacité thermique massique $c_p = 1\,000\ \mathrm{J.kg^{-1}.K^{-1}}$, masse volumique $\rho = 2\,150\ \mathrm{kg.m^{-3}}$.

PARTIE I : ETUDE PRELIMINAIRE

Dans cette partie, on établit l'équation gouvernant les variations de la température et on la résout en régime permanent.

I.A. Equation gouvernant la température

On suppose que la température dans le mur T ne dépend que du temps t et de la coordonnée x.

I.A.1. A quelle condition peut-on supposer que la température ne dépend pas des coordonnées y et z ?

I.A.2. Donner l'équation générale qui décrit le transport de chaleur dans un solide en l'absence de source d'énergie. Comment cette équation se simplifie-t-elle sous les hypothèses de la question I.A.1 ?

I.B. Conditions aux limites

On envisage plusieurs types de conditions aux limites.

(i) La température est imposée aux limites du système.

(ii) La paroi extérieure est isolée par un matériau de très faible conductivité.

I.B.1. Traduire chacune de ces conditions aux limites sur la fonction $T(x,t)$ et/ou sa dérivée.

Dans toute la suite, on adoptera des conditions aux limites de type température imposée.

I.C. Solutions en régime permanent

I.C.1. Résoudre l'équation obtenue à la question I.A.2. en régime permanent, avec les conditions aux limites de type températures imposées (question I.B.(i)) :

- pour un instant particulier négatif $t_1 < 0$,
- pour un instant particulier positif $t_2 > 0$, très longtemps après la variation de température extérieure, quand le régime permanent est de nouveau établi dans le mur.

I.C.2. Quelle est la nature des profils $T(x)$ obtenus (en régime permanent) à ces deux instants ? Tracer à la main les deux profils sur un même graphique sur la copie.

I.C.3. Sur le même graphique, tracer à la main qualitativement les profils intermédiaires à différents instants entre la variation brutale de la température extérieure ($t = 0$) et l'instant t_2 où le régime est de nouveau permanent.

PARTIE II : RESOLUTION NUMERIQUE

II.A. Equation à résoudre

On cherche à résoudre numériquement l'équation aux dérivées partielles :

$$\alpha \frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2} \qquad (1)$$

où α est une constante. A l'équation (1) sont associées les conditions :

$$\begin{aligned} &T(0,t) = T_{int} \quad \text{pour tout } t > 0 \\ &T(e,t) = T_{ext2} \quad \text{pour tout } t > 0 \\ &T(x,0) = ax + b \quad \text{pour tout } x \in [0,e] \end{aligned}$$

II.A.1. Quelle est l'expression de α en fonction des paramètres physiques du mur ?

II.A.2. Exprimer a et b en fonction de T_{int}, T_{ext1} et e.

Pour effectuer la résolution de l'équation (1), nous utiliserons la méthode des différences finies présentée dans la partie II.B.

II.B. Méthode des différences finies

II.B.1. Discrétisation dans l'espace et dans le temps

On divise l'intervalle $[0,e]$, représentant l'épaisseur du mur, en $N+2$ points, numérotés de 0 à $N+1$, régulièrement espacés de Δx (figure 2, page suivante). Cette division est appelée « discrétisation ». La distance Δx est appelée le « pas d'espace ». A l'intérieur du mur (frontières intérieure et extérieure exclues) se trouvent donc N points. On cherche à obtenir la température en ces points particuliers à chaque instant.

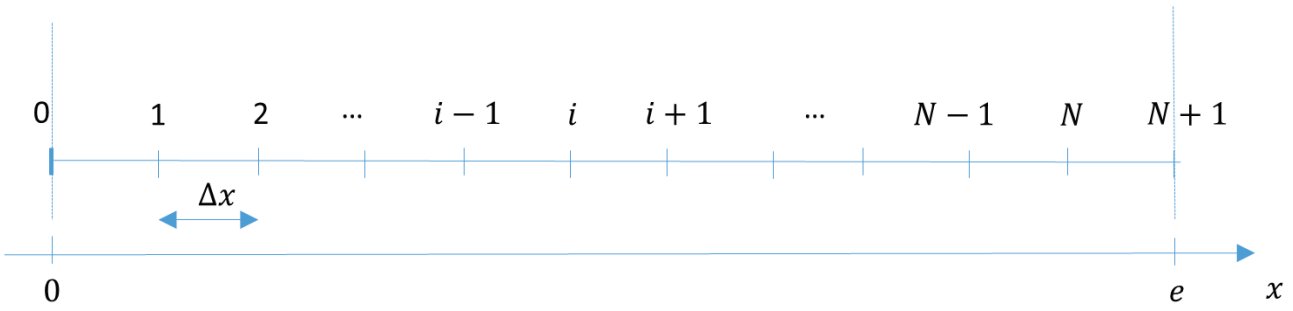


Figure 2 - Discrétisation spatiale dans la direction x.

II.B.1.a. Donner l'expression de Δx en fonction de N et de l'épaisseur du mur e.

II.B.1.b. Donner l'abscisse x_i du i^e point en fonction de i et Δx, sachant que $x_0 = 0$ et $x_{N+1} = e$.

Le temps est discrétisé en $ItMax$ intervalles de durée Δt et on ne s'intéresse au profil de température qu'aux instants particuliers $t_k = k.\Delta t$. L'intervalle élémentaire de temps Δt est appelé le « pas de temps ».

Pour résoudre l'équation (1), deux méthodes sont proposées :

- méthode utilisant un schéma explicite,
- méthode utilisant un schéma implicite.

II.B.2. Méthode utilisant un schéma explicite

II.B.2.a. A l'aide d'un développement limité de la fonction $x \mapsto T(x,t)$, donner une expression de $T(x+\Delta x,t)$ à l'ordre 3 ($o(\Delta x^3)$) en fonction de T et de ses dérivées partielles par rapport à x évaluées en (x,t). De même, donner une expression de $T(x-\Delta x,t)$ à l'ordre 3.

II.B.2.b. En déduire une expression approchée à l'ordre 1 ($o(\Delta x)$) de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ (dérivée partielle spatiale seconde de T évaluée au point x à l'instant t) en fonction de $T(x+\Delta x,t)$, $T(x-\Delta x,t)$ et $T(x,t)$ et Δx.

On note T_i^k la température $T(x_i,t_k)$, évaluée au point d'abscisse x_i à l'instant t_k. De même, on note $T_{i+1}^k = T(x_i+\Delta x,t_k)$ et $T_{i-1}^k = T(x_i-\Delta x,t_k)$.

II.B.2.c. Déduire de la question précédente une expression approchée de $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i,\,t_k}$ (dérivée partielle spatiale seconde de T évaluée en x_i à l'instant t_k) en fonction de T_i^k, T_{i+1}^k et T_{i-1}^k et Δx.

La dérivée partielle temporelle de l'équation (1) est maintenant approchée grâce à un développement limité.

II.B.2.d. A l'aide d'un développement limité de la fonction $t \mapsto T(x,t)$, donner une expression de $T(x,t+\Delta t)$ à l'ordre 1 ($o(\Delta t)$) en fonction de T et de sa dérivée partielle par rapport à t évaluées en (x,t).

II.B.2.e. En déduire une valeur approchée de $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ (dérivée partielle par rapport au temps de T évaluée au point x à l'instant t) à l'ordre 0 ($o(1)$) en fonction de $T(x,t+\Delta t)$, $T(x,t)$ et Δt.

II.B.2.f. Donner une expression de $\left.\frac{\partial T}{\partial t}\right|_{x_i,\, t_k}$ (dérivée partielle par rapport au temps de T évaluée en x_i à l'instant t_k) en fonction de Δt, T_i^k et T_i^{k+1}, avec $T_i^{k+1} = T(x_i, t_k + \Delta t)$.

L'équation (1) est valable en chaque point d'abscisse x_i et à chaque instant t_k.

II.B.2.g. Ecrire la forme approchée de cette équation au point i et à l'instant k en approchant $\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.c. et en approchant $\left.\frac{\partial T}{\partial t}\right|_{x,t}$ avec la formule obtenue à la question II.B.2.f.

II.B.2.h. Montrer que l'équation obtenue à la question II.B.2.g peut s'écrire sous la forme :

$$T_i^{k+1} = rT_{i-1}^k + (1-2r)T_i^k + rT_{i+1}^k \qquad (2)$$

en précisant la valeur du paramètre r en fonction de $\Delta x, \Delta t$ et α.

L'équation (2) est appelée schéma numérique explicite. Si on connaît la température en tous les points $x_1, x_2, \ldots, x_{N-1}, x_N$ à l'instant t_k, on peut calculer grâce à elle la température en tous les points à l'instant ultérieur t_{k+1}.

II.B.2.i. L'équation (2) est-elle valable dans tout le domaine, c'est-à-dire pour toute valeur de $i,\ 0 \leq i \leq N+1$? Que valent T_0^k et T_{N+1}^k ?

II.B.2.j. Dans cette question, on élabore une fonction `schema_explicite` permettant de calculer la température en chaque point au cours du temps selon la formule (2). Parmi les variables d'entrée se trouvera un vecteur `T0` de dimension N, défini en dehors de la fonction, contenant les valeurs de la température aux points de discrétisation à l'instant initial. Au sein de la fonction, un algorithme calculera itérativement la température avec un nombre maximal d'itérations `ItMax`. En sortie de la fonction, on récupérera le nombre d'itérations réellement effectuées, `nbIter` et une matrice `T_tous_k`, de dimensions $N \times ItMax$. Chaque colonne de cette matrice contient le vecteur $\boldsymbol{T^k}$ dont les éléments sont les valeurs de la température aux N points $x_1, \ldots, x_N$ (points à l'intérieur du mur) à l'instant k :

$$\boldsymbol{T^k} = \begin{pmatrix} T_1^k \\ T_2^k \\ \ldots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \qquad \text{et} \qquad \texttt{T\_tous\_k} = \begin{pmatrix} T_1^1 & T_1^2 & \ldots & T_1^k & \ldots & T_1^{k-1} & T_1^k \\ T_2^1 & T_2^2 & & T_2^k & & T_2^{k-1} & T_2^k \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ T_{N-1}^1 & T_{N-1}^2 & \ldots & T_{N-1}^k & \ldots & T_{N-1}^{k-1} & T_{N-1}^k \\ T_N^1 & T_N^2 & & T_N^k & & T_N^{k-1} & T_N^k \end{pmatrix}.$$

On souhaite arrêter le calcul lorsque la température ne varie presque plus dans le temps. Dans ce but, on évaluera la norme 2 de $\boldsymbol{T^k - T^{k-1}}$ à chaque itération. La définition de la norme 2 est rappelée à la question II.B.2.j.*(vi)*.

II.B.2.j.*(i)* Ecrire l'en-tête de la fonction en précisant bien les paramètres d'entrée et de sortie.

II.B.2.j.*(ii)* Le schéma numérique (2) permet d'approcher avec succès la solution à la condition $r < 1/2$. Programmer un test qui avertit l'utilisateur si cette condition n'est pas respectée.

II.B.2.j.*(iii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` de dimensions $N \times ItMax$ en la remplissant de zéros.

II.B.2.j.*(iv)* Remplacer la première colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.2.j.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$, en distinguant le cas $i = 1$, le cas $2 \leq i \leq N - 1$ et le cas $i = N$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.2.j.*(vi)* Ecrire une fonction `calc_norme` qui calcule la norme 2 d'un vecteur. On rappelle que la norme 2 d'un vecteur $\boldsymbol{V}$ s'écrit :

$$\|\boldsymbol{V}\|_2 = \sqrt{\sum_{i=1}^{N} V_i^2} \quad \text{avec} \quad \boldsymbol{V} = \begin{pmatrix} V_1 \\ \vdots \\ V_i \\ \vdots \\ V_N \end{pmatrix}.$$

II.B.2.j.*(vii)* Elaborer une boucle permettant de calculer itérativement le profil de température aux instants $t_k = k.\Delta t$ avec $k \geq 2$. Cette boucle sera interrompue lorsque la norme 2 du vecteur $\boldsymbol{T^k - T^{k-1}}$ deviendra inférieure à 10^{-2} ou lorsque le nombre d'itérations atteindra la valeur $ItMax$ (prévoir les deux cas). Utiliser, pour cela, la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.2.j.*(viii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.B.3. Méthode utilisant un schéma implicite

Le schéma explicite (2) ne converge que si le pas de temps Δt est suffisamment faible par rapport au pas d'espace Δx. Si l'on souhaite effectuer un calcul pour un temps physique long, beaucoup d'itérations seront nécessaires et le temps de calcul sera très long. C'est pourquoi on préfère d'autres types de schémas appelés schémas implicites.

Dans cette partie, la dérivée partielle seconde par rapport à x de la température apparaissant dans l'équation (1) est évaluée au point d'abscisse x_i et à l'instant $k + 1$:

$$\left.\frac{\partial^2 T}{\partial x^2}\right|_{x,t} \approx \left.\frac{\partial^2 T}{\partial x^2}\right|_{x_i, t_{k+1}}$$

et la dérivée partielle par rapport à t est évaluée au point d'abscisse x_i et à l'instant k :

$$\left.\frac{\partial T}{\partial t}\right|_{x,t} \approx \left.\frac{\partial T}{\partial t}\right|_{x_i, t_k}.$$

II.B.3.a. Donner la nouvelle expression approchée de l'équation (1) définie en page 3.

II.B.3.b. Montrer que l'équation obtenue à la question II.B.3.a. peut être mise sous la forme

$$T_i^k = -rT_{i-1}^{k+1} + (1 + 2r)T_i^{k+1} - rT_{i+1}^{k+1} \quad . \qquad (3)$$

L'équation (3) est appelée schéma implicite car la température à l'instant t_k est exprimée en fonction de la température à l'instant ultérieur t_{k+1}.

Le système d'équations ainsi obtenu peut être écrit sous la forme :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + r\,\boldsymbol{v} \qquad (4)$$

où M est une matrice carrée $N \times N$ et $\boldsymbol{T}^{k}$ et $\boldsymbol{T}^{k+1}$ sont les vecteurs de dimension N définis par :

$$\boldsymbol{T}^{k} = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix} \quad \text{et} \quad \boldsymbol{T}^{k+1} = \begin{pmatrix} T_1^{k+1} \\ T_2^{k+1} \\ \dots \\ T_{N-1}^{k+1} \\ T_N^{k+1} \end{pmatrix}$$

et $\boldsymbol{v}$ est un vecteur de taille N faisant intervenir les conditions aux limites.

II.B.3.c. Préciser l'expression de la matrice M et l'expression du vecteur $\boldsymbol{v}$.

A chaque pas de temps, il faut inverser le système matriciel :

$$M\boldsymbol{T}^{k+1} = \boldsymbol{T}^{k} + r\,\boldsymbol{v}$$

pour obtenir $\boldsymbol{T}^{k+1}$à partir de $\boldsymbol{T}^{k}$.

II.B.3.d. Le but de cette question est d'écrire une fonction `CalcTkp1` qui permet de résoudre un système matriciel tridiagonal en utilisant l'algorithme de Thomas présenté ci-dessous.

Algorithme de Thomas :
On cherche à résoudre un système matriciel tridiagonal de la forme :

$$M\,\boldsymbol{u} = \boldsymbol{d} \qquad (5)$$

où M est une matrice de dimensions $N \times N$ tridiagonale, c'est-à-dire une matrice dont tous les éléments sont nuls, sauf sur la diagonale principale, la diagonale supérieure et la diagonale inférieure

$$M = \begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & 0 & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & a_{N-1} & b_{N-1} & c_{N-1} \\ & & & & a_N & b_N \end{pmatrix}$$

et où les vecteurs $\boldsymbol{u}$ et $\boldsymbol{d}$, de dimension N, s'écrivent **:**

$$\boldsymbol{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} \quad \text{et} \quad \boldsymbol{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{N-1} \\ d_N \end{pmatrix}$$

Dans cet algorithme, on calcule d'abord les coefficients suivants :

$$c'_1 = \frac{c_1}{b_1}$$
$$c'_i = \frac{c_i}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N-1$$

et

$$d'_1 = \frac{d_1}{b_1}$$
$$d'_i = \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} \quad \text{pour} \quad i = 2, 3, \dots, N\,.$$

Les inconnues $u_1, u_2, \dots, u_N$ sont alors obtenues par les formules :

$$u_N = d'_N$$
$$u_i = d'_i - c'_i u_{i+1} \quad \text{pour} \quad i = N-1, N-2, \dots, 2, 1\ .$$

II.B.3.d.*(i)* En utilisant l'algorithme de Thomas, écrire une fonction `CalcTkp1` qui permet de calculer le vecteur $\boldsymbol{u}$, solution du système matriciel (5), à partir de la matrice *M* et du vecteur $\boldsymbol{d}$.

II.B.3.e. Dans cette question, une fonction `schema_implicite` est élaborée avec les mêmes arguments d'entrée et de sortie que la fonction `schema_explicite` (définis à la question II.B.2.j.**)** et qui utilise les mêmes critères d'arrêt (définis à la question II.B.2.j.*(vii)*).

II.B.3.e.*(i)* Ecrire l'en-tête de la fonction en précisant les paramètres d'entrée et de sortie.

II.B.3.e.*(ii)* Affecter la valeur 2 000 à `ItMax`. Créer la matrice `T_tous_k` dont les dimensions sont $N \times ItMax$ en la remplissant de zéros.

II.B.3.e.*(iii)* Remplacer la 1re colonne de `T_tous_k` par le vecteur des valeurs initiales `T0`.

II.B.3.e.*(iv)* Définir la matrice `M` et le vecteur `v` qui interviennent dans l'équation (4).

II.B.3.e.*(v)* Calculer le profil de température à l'instant $k = 1$ $(t = \Delta t)$. Affecter ces valeurs à la deuxième colonne de `T_tous_k`.

II.B.3.e.*(vi)* Ecrire une boucle permettant de calculer itérativement le profil de température aux instants ultérieurs $t_k = k \times \Delta t$ avec $k \geq 2$, en prévoyant un arrêt lorsque la norme 2 du vecteur $\boldsymbol{T^k - T^{k-1}}$ devient inférieure à 10^{-2} ou lorsque le nombre d'itérations atteint la valeur `ItMax` (prévoir les deux cas). Utiliser pour cela la fonction `calc_norme` définie à la question II.B.2.j.*(vi)*.

II.B.3.e.*(vii)* Ecrire la fin de la fonction afin de renvoyer tous les arguments de sortie définis au début de la question II.B.2.j.

II.C. Programme principal

II.C.1. Début du programme

II.C.1.a. Définir les variables `epais` (épaisseur du mur), `conduc` (conductivité thermique), `rho` (masse volumique), `Cp` (capacité thermique massique), `Tint` (température intérieure), `Text1`

(température extérieure pour les instants $t < 0$), `Text2` (température extérieure pour les instants $t > 0$), `N` (nombre de points de calcul **à l'intérieur du mur**) et `Dt` (intervalle de temps élémentaire) et leur affecter les valeurs correspondant au problème physique défini au début de l'énoncé. On prendra un nombre de points de discrétisation $N = 60$ et un pas de temps Δt de 25 secondes.

II.C.1.b. Calculer les coefficients `a` et `b` avec la formule trouvée à la question II.A.2.

II.C.1.c. Créer un vecteur `x` dont les éléments $x_1, x_2, \ldots, x_N$ sont définis à la question II.B.1.b.

II.C.1.d. Calculer le vecteur des températures initiales `T0`.

II.C.1.e. Calculer `alpha` selon la formule trouvée à la question II.A.1. Calculer `r` en utilisant la formule calculée à la question II.B.2.h.

II.C.2. Calcul des températures

II.C.2.a. Ecrire un morceau de programme qui demande à l'utilisateur quel schéma (explicite ou implicite) il souhaite utiliser et qui appelle la fonction correspondante.

II.C.3. Analyse du résultat

II.C.3.a. Ecrire un morceau de programme permettant de tracer sur un même graphique le profil de température en fonction de `x` tous les 100 pas de temps.

II.C.3.b. Faire afficher le temps en heures au bout duquel le régime permanent est établi.

Fin de l'énoncé

ANNEXE A : COMMANDES ET FONCTIONS USUELLES DE SCILAB

A=[a b c d;e f g h;i j k l]

Description : commande permettant de créer une matrice dont la première ligne contient les éléments a, b, c, d, la seconde ligne contient les éléments e, f, g, h et la troisième, les éléments i, j, k, l.

Exemple : A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]

⇨
```
 1.   2.   3.   4.   5.
 3.   10.  11.  12.  20.
 0.   1.   0.   0.   2.
```

A(i,j)

Arguments d'entrée : les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément (i, j) de la matrice A.

Description : fonction qui retourne l'élément (i, j) de la matrice A. Pour obtenir toute la colonne j de la matrice A, on utilise la syntaxe $A(:, j)$. De même, pour accéder à l'intégralité de la ligne i de la matrice A, on écrit $A(i, :)$.

Exemple :
```
A=[1 2 3 4 5;3 10 11 12 20;0 1 0 0 2]
A(2,4)
⇨ 12
A(:,3)
⇨  3.
   11.
   0.
A(2,:)
⇨ 3.   10.   11.   12.   20.
```

x=[x1:Dx:x2]

Description : commande permettant de créer un vecteur dont les éléments sont espacés de Dx et dont le premier élément est x_1 et le dernier élément est le plus grand multiple de Dx inférieur ou égal à x_2.

ATTENTION : le vecteur ainsi créé est un vecteur ligne. Pour convertir un vecteur ligne en un vecteur colonne, on le transpose en utilisant l'apostrophe « ' » : x_trans=x'.

Exemple *:*
```
x=[2:0.5:6.3]
⇨ 2.   2.5   3.   3.5   4.   4.5   5.   5.5   6.
x_trans=x'
⇨ 2.
  2.5
  3.
  3.5
  4.
  4.5
  5.
  5.5
  6.
```

zeros(n,m)

Arguments d'entrée : deux entiers n et m correspondant aux dimensions de la matrice à créer.

Argument de sortie : un tableau (matrice) d'éléments nuls.

Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.

Exemple : zeros(3,4)

⇨ 0. 0. 0. 0.
0. 0. 0. 0.
0. 0. 0. 0.

plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).

Description : fonction permettant de tracer sur un graphique n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y.

Exemple : x= [3:0.1:5]
y=sin(x)
plot(x,y)

ANNEXE B : BIBLIOTHEQUE NUMPY DE PYTHON

Dans les exemples ci-dessous, la bibliothèque numpy a préalablement été importée à l'aide de la commande : **import numpy as np**

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

np.array(liste)

Argument d'entrée : une liste définissant un tableau à 1 dimension (vecteur) ou 2 dimensions (matrice).

Argument de sortie : un tableau (matrice).

Description : fonction permettant de créer une matrice (de type tableau) à partir d'une liste.

Exemple : np.array([4,3,2])
⇨ [4 3 2]
np.array([[5],[7],[1]])
⇨ [[5]
[7]
[1]]
np.array([[3,4,10],[1,8,7]])
⇨ [[3 4 10]
[1 8 7]]

$\boldsymbol{A[i,j]}$.

Arguments d'entrée : un tuple contenant les coordonnées de l'élément dans le tableau A.

Argument de sortie : l'élément $(\mathrm{i}+1, \mathrm{j}+1)$ de la matrice A.

Description : fonction qui retourne l'élément $(i+1, j+1)$ de la matrice A. Pour obtenir toute la colonne j+1 de la matrice A, on utilise la syntaxe $A[:, j]$. De même, pour accéder à l'intégralité de la ligne i+1 de la matrice A, on écrit $A[i, :]$.

ATTENTION : en langage Python, les lignes d'un A de dimension $n \times m$ sont numérotées de 0 à $n-1$ et les colonnes sont numérotées de 0 à $m-1$

Exemple : A=np.array([[3,4,10],[1,8,7]])
A[0,2]
⇨ 10
A[:,2]
⇨ [10 7]
A[1,:]
⇨ [1 8 7]

np.zeros((n,m))

Arguments d'entrée : un tuple de deux entiers correspondant aux dimensions de la matrice à créer.

Argument de sortie : un tableau (matrice) d'éléments nuls.

Description : fonction créant une matrice (tableau) de dimensions $n \times m$ dont tous les éléments sont nuls.

Exemple : np.zeros((3,4))

⇨ [[0 0 0 0]
[0 0 0 0]
[0 0 0 0]]

np.linspace(Min,Max,nbElements)

Arguments d'entrée : un tuple de 3 entiers.

Argument de sortie : un tableau (vecteur).

Description : fonction créant un vecteur (tableau) de $nbElements$ nombres espacés régulièrement entre Min et Max. Le 1er élément est égal à Min, le dernier est égal à Max et les éléments sont espacés de $(Max - Min)/(nbElements - 1)$:

Exemple : np.linspace(3,25,5)

⇨ [3 8.5 14 19.5 25]

ANNEXE C : BIBLIOTHEQUE MATPLOTLIB.PYPLOT DE PYTHON

Cette bibliothèque permet de tracer des graphiques. Dans les exemples ci-dessous, la bibliothèque matplotlib.pyplot a préalablement été importée à l'aide de la commande :
import matplotlib.pyplot as plt

On peut alors utiliser les fonctions de la bibliothèque, dont voici quelques exemples :

plt.plot(x,y)

Arguments d'entrée : un vecteur d'abscisses x (tableau de dimension n) et un vecteur d'ordonnées y (tableau de dimension n).

Description : fonction permettant de tracer sur un graphique de n points dont les abscisses sont contenues dans le vecteur x et les ordonnées dans le vecteur y. Cette fonction doit être suivie de la fonction **plt.show()** pour que le graphique soit affiché.

Exemple :
```
x= np.linspace(3,25,5)
y=sin(x)
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

plt.xlabel(nom)

Argument d'entrée : une chaîne de caractères.

Description : fonction permettant d'afficher le contenu de nom en abscisse d'un graphique.

plt.ylabel(nom)

Argument d'entrée : une chaîne de caractères.

Description : fonction permettant d'afficher le contenu de nom en ordonnée d'un graphique.

plt.show()

Description : fonction réalisant l'affichage d'un graphe préalablement créé par la commande **plt.plot(x,y)**. Elle doit être appelée après la fonction plt.plot et après les fonctions plt.xlabel et plt.ylabel.