504

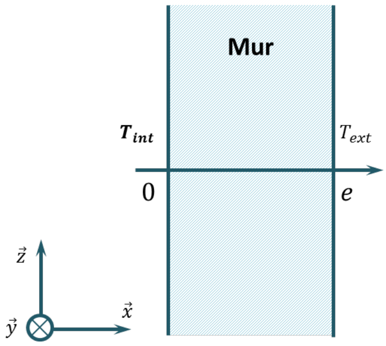
Transfert thermique dans un mur

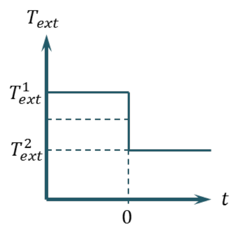
Savoirs et compétences :
- *Savoir utiliser une bibliothèque de calcul scientifique.*

Mise en situation

On étudie les transferts thermiques dans le mur d'une maison. La température à l'intérieur de la maison est constante dans le temps et égale à $T_{\text{int}} = 20°\text{C}$. Aux temps négatifs $t < 0$, la température extérieure est égale à $T_{\text{ext},1} = 10°\text{C}$. À $t = 0$, elle chute brusquement à $T_{\text{ext},2} = -10°C$ et elle reste égale à cette valeur aux temps positifs ($t > 0$). On souhaite étudier l'évolution du profil de température dans le mur au cours du temps.

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





On suppose que les longueurs L_y et L_z suivant $\overrightarrow{y}$ et $\overrightarrow{z}$ sont très grandes devant l'épaisseur e. En conséquence, on suppose que la température T dans le mur ne dépend que du temps t et de la coordonnée x.

Objectif L'objectif est de déterminer l'évolution du flux thermique dans le mur au cours du temps. Pour cela, on s'appuiera sur la résolution d'une équation différentielle en utilisant un schéma explicite puis implicite.

Équation gouvernant la température

En utilisant les hypothèses dimensionnelles, l'équation de la chaleur simplifiée es donnée par

$$\frac{\rho\, c_p}{\lambda}\frac{\partial T(x,t)}{\partial t} = \frac{\partial^2 T(x,t)}{\partial x^2}.$$

On envisage le cas où la température est imposée aux limites du système c'est-à-dire T_{int} et T_{ext} sont imposées. On a donc $T(0,t) = T_{\text{int}}$ et $T(e,t) = T_{\text{ext}}$

En régime permanent dans les conditions suivantes, on a :
- pour un instant particulier négatif $t_1 < 0$, $T(x) = \dfrac{T_{\text{ext},1} - T_{\text{int}}}{e} x + T_{\text{int}}$;
- 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, $T(x) = \dfrac{T_{\text{ext},2} - T_{\text{int}}}{e} x + T_{\text{int}}$.

Question 1 *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.*

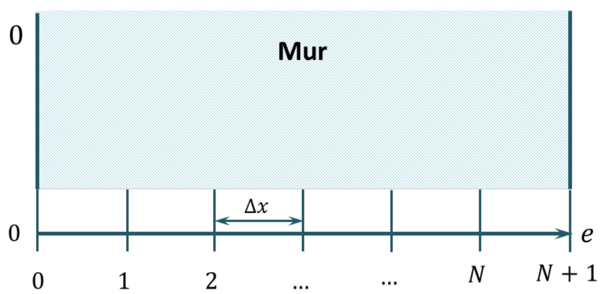
Résolution numérique : méthode des différences finies

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

$$\frac{\rho\,\mathbf{c_p}}{\lambda}\frac{\partial \mathbf{T(x,t)}}{\partial \mathbf{t}} = \frac{\partial^2 \mathbf{T(x,t)}}{\partial \mathbf{x^2}}. \qquad (1)$$

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. 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.



On a : $\Delta x = \dfrac{e}{N+1}$ et $x_i = i\Delta x$.

Le temps est discrétisé en *ItMax* intervalles de durée Δt et on ne s'intéresse au profil de température qu'aux

instants particuliers $t_k = k\cdot\Delta t$. L'intervalle élémentaire de temps Δt est appelé le « pas de temps ».
Deux méthodes de résolutions sont proposées :

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

Méthode utilisant un schéma explicite

Objectif Déterminer le schéma explicite permettant la résolution de l'équation de la chaleur.

L'expression approchée à l'ordre 1 de $\left[\dfrac{\partial^2 T(x,t)}{\partial x^2}\right]_{x,t}$ est donnée par
$\dfrac{\partial^2 T(x,t)}{\partial x^2} = \dfrac{T(x+\Delta x,t)-2T(x,t)+T(x-\Delta x,t)}{\Delta x^2} + o(\Delta x)$.

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 $T_{i-1}^k = T(x_i-\Delta x,t_k)$.

Ainsi

$$\left[\frac{\partial^2 T(x,t)}{\partial x^2}\right]_{x_i,t_k} = \frac{T_{i+1}^k - 2T_i^k + T_{i-1}^k}{\Delta x^2}.$$

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

En utilisant le même raisonnement en réalisant un développement limité de la fonction $t\mapsto T(x,t)$ à l'ordre 1, on obtiendrait l'équation suivante valable en chaque point d'abscisse x_i et à chaque instant t_k :

$$\left[\frac{\partial T(x,t)}{\partial t}\right]_{x_i,t_k} = \frac{T_i^{k+1}-T_i^k}{\Delta t}.$$

Question 2 *En utilisant les questions précédentes, montrer que l'équation différentielle peut se mettre sous la forme :*

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

r sera explicité en fonction de Δx, Δt et α.

Correction $r = \dfrac{\Delta t}{\alpha \Delta x^2}$

Question 3 *L'équation est-elle valable dans tout le domaine, c'est-à-dire pour toute valeur de i, $0\le i\le N+1$? Que valent T_0^k et T_{N+1}^k ?*

Correction Oui...
On a : $T_0^k = 20^o C$ et $T_{N+1}^k = -10^o C$.

Question 4 *Montrer que pour tout instant k, le problème peut se mettre sous la forme matricielle suivante :*

$$T^{k+1} = M\cdot T^k + rV \quad \text{avec} \quad T^k = \begin{pmatrix} T_1^k \\ T_2^k \\ \dots \\ T_{N-1}^k \\ T_N^k \end{pmatrix}. \tag{0.1}$$

avec M une matrice carrée $N\times N$, V un vecteur de taille N que l'on explicitera.

Correction On a, pour $i\in]0;N+1[$:

i	$T_i^{k+1} = rT_{i-1}^k + (1-2r)T_i^k + rT_{i+1}^k$
$i=1$	$T_1^{k+1} = rT_0^k + (1-2r)T_1^k + rT_2^k$
$i=2$	$T_2^{k+1} = rT_1^k + (1-2r)T_2^k + rT_3^k$
$i=3$	$T_3^{k+1} = rT_2^k + (1-2r)T_3^k + rT_4^k$
$i=N-1$	$T_{N-1}^{k+1} = rT_{N-2}^k + (1-2r)T_{N-1}^k + rT_N^k$
$i=N$	$T_N^{k+1} = rT_{N-1}^k + (1-2r)T_N^k + rT_{N+1}^k$

On a donc :

$$T^{k+1} = M\cdot T^k + rV$$

avec :

$$M = \begin{pmatrix} 1-2r & r & 0 & 0 & 0 & \dots & 0 \\ r & 1-2r & r & 0 & 0 & \dots & 0 \\ 0 & r & 1-2r & r & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \dots \\ 0 & 0 & 0 & 0 & 0 & r & 1-2r \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} T_0^k = \\ 0 \\ \vdots \\ 0 \\ T_N^k = \end{pmatrix}$$

Question 5 *Expliciter succinctement comment déterminer la température dans le mur à chaque instant.*

Correction À l'instant $t=0$, le flux de température dans le mur est connu. La température extérieure passe alors à $-10^o C$. On utilise alors l'équation de récurrence.

(Lors du codage de la méthode explicite, il faut faire attention à la divergence du schéma. Pour cela, il faut prendre des intervalles de temps « petits ».)

Question 6 *On donne* `T0` *le vecteur température à l'instant $k=0$. Écrire la fonction* `euler_explicite(M,T0,V,k)` *retournant le vecteur de température T^k à l'instant k. Cette fonction sera définie de manière* ***récursive****.*

Correction

```
def euler_explicite(M,T_0,r,V,k):
    if k==1:
        return np.dot(M,T_0)+r*V
    else:
        T=euler_explicite(M,T_0,r,V,k-1)
        return np.dot(M,T)+r*V
```

Méthode utilisant un schéma implicite

Question 7 *Pour obtenir T^{k+1} en fonction de T^k, il est nécessaire d'inverser le système matriciel à chaque pas de temps. Donner le nom d'un algorithme permettant de faire cela et donner sa complexité.*

Correction On peut utiliser l'algorithme du pivot de Gauss. Sa complexité est en $\mathcal{O}(n^3)$.

Question 8 *En utilisant l'algorithme de Thomas, écrire une fonction* `CalcTkp1(M,D)` *qui retourne le vecteur U, solution du système matriciel, à partir de la matrice M et du vecteur D.*

On pourra commencer par créer des vecteurs contenant des zéros : A, B, C, Cp ,Dp (pour C' et D') et U. Puis on pourra définir A, B et C à partir de la matrice M et ensuite calculer Cp et Dp puis U.

Correction
Proposition de corrigé qui ne correspond pas exactement aux préconisations de l'énoncé :

```
def CalcTkp1(M,d):
    N=d.shape[0]
    c=[M[0,1]/M[0,0]]
    d_prime=[d[0,0]/M[0,0]]
    u=np.zeros((N,1))
    for i in range(1,N-1):
        c.append(M[i,i+1]/(M[i,i]-M[i,i-1]*c[i-1]))
        d_prime.append((d[i,0]-M[i,i-1]*d_prime[-1])/(
    d_prime.append((d[N-1,0]-M[N-1,N-2]*d_prime[-1])/(
    u[N-1,0]=d_prime[-1]
    for j in range(2,N):
        u[N-j,0]=d_prime[N-j]-c[N-j]*u[N-j+1,0]
    return u
```

Question 9 *Donner la complexité de l'algorithme et comparer à celui de la question 16. On prendra en compte uniquement les tests, les affectations (même de tableaux) et les opérations élémentaires (+, -, *, /) qui compte chacune pour « un ».*

Correction La complexité de l'algorithme est linéaire (à comparer à une complexité cubique pour l'algorithme de Gauss).

1 Résolution de l'équation différentielle implicite

Question 10 *Écrire une fonction* `calc_norme` *qui calcule la norme d'un vecteur.*

Correction

```
def calc_norme(v):
    res = 0
    for i in range(len(v)):
        res = res+v[i]**2
    return math.sqrt(res)
```

Question 11 *Écrire la fonction* `solution(M,T,V)`, *d'arguments une matrice M tridiagonale et deux vecteurs T et V et retournant le vecteur U tels que $MU = T + rV$.* On utilisera la fonction `CalcTkp1`.

Correction

```
def solution(M,T,V) :
    """
    Entrées :
        * M, np.array (N+1 x N+1) : matrice à
inverser
        * T, np.array (N+1 x 1) : température à
chaque abscisse du mur àl'instant k
        * V, np.array (N+1 x 1) : température *imposée* à
chaque abscisse du mur àl'instant final
    Sortie :
        * TT, np.array (N+1 x 1) : température à
chaque abscisse du mur àl'instant k+1
    """
    # On considère que r a été définie auparavant comme variable
    D = T+r*V
    return CalcTkp1(M,D)
```

Question 12 *Affecter la valeur 2000 à* `ItMax`. *Créer la matrice* `T_tous_k` *de dimensions $N \times ItMax$ en la remplissant de zéros.*

Correction

```
ItMax = 2000
T_tous_k = np.zeros([N,ItMax])
```

Question 13 *D'après les questions précédentes la température T dans le mur vérifie la condition initiale $T(x,0) = ax + b$ où $a = \dfrac{T_{ext,1} - T_{int}}{e}$ et $b = T_{int}$.*
Écrire la fonction `T0(Tint,Text,N)` *d'arguments la température intérieure T_{int}, la température extérieure T_{ext} et l'entier N et retournant le vecteur T^0, de taille N, contenant les températures en chaque point du mur lorsque* $t = 0$.

Correction On a : $T(x_i, k=0) = \dfrac{T_{\text{ext},1} - T_{\text{int}}}{e} x_i + T_{\text{int}} = \dfrac{T_{\text{ext},1} - T_{\text{int}}}{e} \cdot i \cdot \dfrac{e}{N+1} + T_{\text{int}} = \dfrac{T_{\text{ext},1} - T_{\text{int}}}{N+1} \cdot i + T_{\text{int}}$.

```
def T0(Ti,Tf,N):
    tt = np.zeros((N,1))
    for i in range(N):
        tt[i] = ((Tf-Ti)/(N-1))*i+Ti #si il y a N points on a di
    return tt
```

Question 14 *Écrire la suite d'instructions qui affecte à la première colonne de* `T_tous_k` *le vecteur des valeurs initiales T^0.* On utilisera la fonction `T0`.

Correction

```
t0 = T0(Tint,Text1,N)
for i in range(N):
    T_tous_k[i][0] = t0[i]
```

Question 15 *Écrire les instructions permettant de définir M et le vecteur V qui interviennent dans l'équation* **??**.

Correction

```
M = np.zeros((N,N))
M[0][0]=1+2*r
M[0][1]=-r

M[N-1][N-1]=1+2*r
M[N-1][N-2]=-r
for i in range(1,N-1):
    M[i][i]=1+2*r
    M[i][i+1]=-r
    M[i][i-1]=-r

V = np.zeros((N,1))
V[0][0]=Tint
V[N-1][0]=Text2
```

2 Analyse des résultats

Question 16 *Donner la suite d'instructions permettant de calculer le profil de la température à l'instant $k=1$ ($t=\Delta t$). C'est à dire le vecteur T^1* . On utilisera la fonction `solution`.

Correction `T1=solution(M,T0,V)`

Question 17 *Écrire la fonction* `euler_implicite(M,T0,V)` *d'argument une matrice M et deux vecteurs $T0$ et V et renvoyant la matrice* `T_tous_k`.
Cette boucle sera interrompue lorsque la norme du vecteur $T^k - T^{k-1}$ deviendra inférieure à 10^{-2} ou lorsque le nombre d'itérations atteindra la valeur ItMax (prévoir deux cas). Utiliser pour cela les fonctions `calc_norme` *et* `solution` *définies précédemment.*

Correction Bloc d'instructions à intégrer dans la fonction `euler_implicite(M,T0,V)` :

```
v=np.zeros((T_tous_k.shape[0],1))
for i in range(T_tous_k.shape[0]):
    v[i,0]=T_tous_k[i,1]-T_tous_k[i,0]
k=1
T=T0(Tint,Text,N)
while calc_norme(v)>10**(-2) and k!=ItMax:
    k=k+1
    T=solution(M,T,V)
    for i in range(T_tous_k.shape[0]):
        T_tous_k[i,k]=T[i,0]
        v[i,0]=T_tous_k[i,k]-T_tous_k[i,k-1]
```

Question 18 *Le pas de discrétisation temporel est de 30 secondes. Les résultats de la simulation sont stockés dans la matrice* `T_tous_k` *définie précédemment. Écrire les instructions permettant de tracer le réseau de courbes précédentes.*

Correction

```
k=[0,240,480,720,960,1200,1440]
les_X=[i*0.4/(N+1) for i in range(N+2)]
for i in k:
    plt.plot(les_X,T_tous_k[:,i])
plt.show()
```

Question 19 *Quelle sera la taille du fichier texte généré ?*

Correction Si on considère que $Itmax = 2000$, on aura donc 100×2000 valeurs. Pour l'estimation de la taille du fichier on fait ici l'abstraction du codage des espaces et des retours à la ligne. La taille du fichier est donc : $100 \times 2000 \times 10 \times 1 \simeq 2$ Mo.

Ch. 4