Sujet : CCP Modélisation PC 2015
Corrigé

Première partie
Étude préliminaire

I.A. Équation gouvernant la température

I.A.1 Si L_y et L_z sont les dimensions caractéristiques selon les axes y et z, alors on peut supposer que la température ne dépend que du temps t et de la coordonnée x si $L_y \gg e$ et $L_z \gg e$.

I.A.2 L'équation de la chaleur sans terme de production peut s'écrire sous la forme suivante. $\rho \cdot c_p \cdot \frac{\partial T}{\partial t} = \lambda \cdot \Delta T \;\Rightarrow\; \rho \cdot c_p \cdot \frac{\partial T}{\partial t} = \lambda \cdot \frac{\partial^2 T}{\partial t^2}$.

I.B. Conditions aux limites

I.B.1 $T(0,t) = T_{int}$ et $T(e,t) = T_{ext}$ si la température est imposée aux limites du système.
$\frac{\partial T}{\partial t}(x,t)_{x=e} = 0$ si la paroi extérieure est isolée par un matériau de très faible conductivité.

I.C. Solutions en régime permanent

I.C.1 L'équation obtenue devient en régime permanent $\frac{\partial^2 T}{\partial t^2} = 0$, donc une évolution de la température linéaire.

- Pour $t_1 < 0$, $T(x) = \frac{T_{ext1} - T_{int}}{e} \cdot x + T_{int}$.
- Pour $t_2 > 0$, $T(x) = \frac{T_{ext2} - T_{int}}{e} \cdot x + T_{int}$.

I.C.2 & I.C.3 Voir courbe figure 1

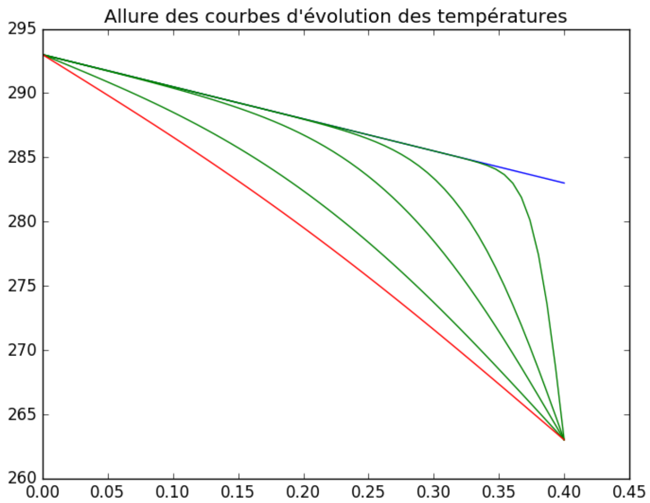

FIGURE 1 – Allures des courbes de température dans la paroi

Deuxième partie

Résolution numérique

II.A. Équation à résoudre

II.A.1 $\alpha = \frac{\rho \cdot c_p}{\lambda}$.

II.A.2 $a = \frac{T_{ext1} - T_{int}}{e}$ et $b = T_{int}$.

II.B. Méthode des différences finies

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

II.B.1.a. $\Delta x = \frac{e}{N+1}$.

II.B.1.b. $x_i = i \times \Delta x$.

II.B.2. Méthode utilisant un schéma explicite

II.B.2.a. $T(x + \Delta x, t) = T(x,t) + \frac{\Delta x^1}{1!} \cdot \frac{\partial T}{\partial t}(x,t) + \frac{\Delta x^2}{2!} \cdot \frac{\partial^2 T}{\partial t^2}(x,t) + \frac{\Delta x^3}{3!} \cdot \frac{\partial^3 T}{\partial t^3}(x,t) + \wr(\Delta x^3)$.
$T(x - \Delta x, t) = T(x,t) - \frac{\Delta x^1}{1!} \cdot \frac{\partial T}{\partial t}(x,t) + \frac{\Delta x^2}{2!} \cdot \frac{\partial^2 T}{\partial t^2}(x,t) - \frac{\Delta x^3}{3!} \cdot \frac{\partial^3 T}{\partial t^3}(x,t) + \wr(\Delta x^3)$.

II.B.2.b. $\frac{\partial^2 T}{\partial t^2}(x,t) = \frac{T(x+\Delta x,t) + T(x-\Delta x,t) - 2 \cdot T(x,t)}{\Delta x^2} + \wr(\Delta x)$.

II.B.2.c. $\frac{\partial^2 T}{\partial t^2}(x_i, t_k) \approx \frac{T_{i+1}^k + T_{i-1}^k - 2 \cdot T_i^k}{\Delta x^2}$.

II.B.2.d. $T(x, t + \Delta t) = T(x,t) + \frac{\Delta t^1}{1!} \cdot \frac{\partial T}{\partial t}(x,t) + \wr(\Delta t)$.

II.B.2.e. $\frac{\partial T}{\partial t}(x,t) = \frac{T(x,t+\Delta t) - T(x,t)}{\Delta t} + \wr(1)$.

II.B.2.f. $\frac{\partial T}{\partial t}(x_i, t_k) \approx \frac{T_i^{k+1} - T_i^k}{\Delta t}$.

II.B.2.g. $\frac{\alpha}{\Delta t} \cdot \left(T_i^{k+1} - T_i^k\right) = \frac{1}{\Delta x^2} \cdot \left(T_{i+1}^k + T_{i-1}^k - 2 \cdot T_i^k\right)$.

II.B.2.h. $T_i^{k+1} = r \cdot \left(T_{i+1}^k + T_{i-1}^k\right) + (1 - 2 \cdot r) \cdot T_i^k$, avec $r = \frac{\Delta t}{\alpha \cdot \Delta x^2}$.

II.B.2.i. L'équation précédente est valable pour $0 < i < N+1$, et $T_0^k = T_{int}$ et $T_{N+1}^k = T_{ext}$.

II.B.2.j. Voici la fonction complète, les parties correspondant aux questions du sujet sont closes avec un commentaire avec la référence de la question tel que `# (vii)`.

```
def schema_explicite(T0, ItMax = 2000, Dt = 25, N = 60, e = 40e-2, \
    Lambda = 1.65, Rho = 2150, c = 1000, T_int = 293, T_ext = 263, \
    epsilon = 1e-2):
    '''renvoie le nombre d'iterations effectuees et une matrice de N lignes
contenant les temperatures a l'instant k en chaque point declare de la paroi,
par la methode des differences finies en utilisant un schéma explicite.'''
    Alpha = Rho * c / Lambda
    Dx = e / (N + 1)
    r = Dt / (Alpha * pow(Dx, 2))

    if r >= 0.5 :
        raise Exception('''r = Dt / ( (rho * c / lambda) * Dx^2 ) doit etre
inferieur à 0.5.''') # Un simple print était possible

    T_tous_k = np.zeros((N, ItMax))

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

```python
    T_tous_k[0, 1] = r * T_int + (1 - 2 * r) * T_tous_k[0, 0] + \
                     r * T_tous_k[1, 0]
    for indice in range(1, N - 1):
        T_tous_k[indice, 1] = r * T_tous_k[indice - 1, 0] + \
                              (1 - 2 * r) * T_tous_k[indice, 0] + \
                              r * T_tous_k[indice + 1, 0]
    T_tous_k[N - 1, 1] = r * T_tous_k[N - 2, 0] + \
                         (1 - 2 * r) * T_tous_k[N - 1, 0] + \
                         r * T_ext

    k = 2
    condition = True

    while condition and k < ItMax - 1 :
        T_tous_k[0, k] = r * T_int + (1 - 2 * r) * T_tous_k[0, k - 1] + \
                         r * T_tous_k[1, k - 1]
        for indice in range(1, N - 1):
            T_tous_k[indice, k] = r * T_tous_k[indice - 1, k - 1] + \
                                  (1 - 2 * r) * T_tous_k[indice, k - 1] + \
                                  r * T_tous_k[indice + 1, k - 1]
        T_tous_k[N - 1, k] = r * T_tous_k[N - 2, k - 1] + \
                             (1 - 2 * r) * T_tous_k[N - 1, k - 1] + \
                             r * T_ext
        if calc_norme(T_tous_k[:, k] - T_tous_k[:, k - 1]) < epsilon :
            condition = False
        k += 1  # nbIter = k - 1

    return (k - 1, T_tous_k)

def calc_norme(V): # (vi)
    '''renvoie la norme 2 du vecteur V de type array. La norme 2 vaut
sqr ( somme (i de 1 à N) Vi^2 ).'''
    return( pow( sum( [pow(vi, 2) for vi in V] ), 0.5 ) )
```

II.B.3. Méthode utilisant un schéma implicite

II.B.3.a. $\alpha \cdot \frac{\partial T}{\partial t}(x_i, t_k) = \frac{\partial^2 T}{\partial x^2}(x_i, t_{k+1})$, c'est à dire $\frac{\alpha}{\Delta t} \cdot \left(T_i^{k+1} - T_i^k\right) = \frac{1}{\Delta x^2} \cdot \left(T_{i+1}^{k+1} + T_{i-1}^{k+1} - 2 \cdot T_i^{k+1}\right)$.

II.B.3.b. $T_i^k = -r \cdot \left(T_{i+1}^{k+1} + T_{i-1}^{k+1}\right) + (1 - 2 \cdot r) \cdot T_i^{k+1}$, avec $r = \frac{\Delta t}{\alpha \cdot \Delta x^2}$.

II.B.3.c. $M = \begin{pmatrix} 1+2\cdot r & -r & 0 & 0 & \cdots & 0 & 0 \\ -r & 1+2\cdot r & -r & 0 & \cdots & 0 & 0 \\ 0 & -r & 1+2\cdot r & -r & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -r & 1+2\cdot r \end{pmatrix}$ et $v = \begin{pmatrix} T_{int} \\ 0 \\ 0 \\ \vdots \\ T_{ext} \end{pmatrix}$.

II.B.3.d. Cette question cachait finalement la fonction la plus longue à écrire.

```python
def CalcTkp1 (M, d) :
    '''renvoie le vecteur u tel que M.u = d selon l'algorithme de Thomas.
M est une matrice tridiagonale de type array,
d est un vecteur colonne de type array.
Le vecteur u renvoye est aussi un vecteur colonne de type array.'''
    N = len(d)   # Nombre de lignes de d, donc sa dimension
    cprime = np.zeros(N - 1)
    cprime[0] =M[0, 1] / M[0,0]    # Attention c'1 = cprime[0]
    for i in range(1, N - 1):    # coherence avec les i - 1 du sujet
        cprime[i] =M[i, i + 1] / (M[i, i] - \
                                  M[i, i - 1] * cprime[i - 1])
    dprime = np.zeros(N)
    dprime[0] = d[0, 0] / M[0,0]    # Attention d'1 = dprime[0]
    for i in range(1, N):
        dprime[i] = (d[i, 0] - M[i, i - 1] * dprime[i - 1]) / \
                    (M[i, i] - M[i, i - 1] * cprime[i - 1])

    u = np.zeros((N, 1))     # u est un vecteur ligne avec u1 = u[0]
    u[N - 1, 0] = dprime[N - 1] # le rang N- 1 correspond au rang N du sujet
    for i in range(N - 2, - 1, -1):    # decroissance de i
        u[i, 0] = dprime[i] - cprime[i] * u[i + 1, 0]

    return (u)
```

II.B.3.e. Le programme suivant est très inspiré bien sûr de celui utilisant le schéma explicite.

```
def schema_implicite(T0, ItMax = 2000, Dt = 25, N = 60, e = 40e-2, \
    Lambda = 1.65, Rho = 2150, c = 1000, T_int = 293, T_ext = 263, \
    epsilon = 1e-2):
    '''renvoie le nombre d'iterations effectuees et une matrice de N lignes
contenant les temperatures a l'instant k en chaque point declare de la paroi,
par la methode des differences finies en utilisant un schéma implicite.'''
    Alpha = Rho * c / Lambda
    Dx = e / (N + 1)
    r = Dt / (Alpha * pow(Dx, 2))

    if r >= 0.5 :
        raise Exception('''r = Dt / ( (rho * c / lambda) * Dx^2 ) doit etre
inferieur à 0.5.''')

    T_tous_k = np.zeros((N, ItMax))

    for i in range(N):
        T_tous_k[i, 0] = T0[i, 0]

    # Definition de M
    M = np.zeros((N, N))

    M[0, 0], M[0, 1] = 1. + 2. * r, -r
    for ligne in range(1, N - 1) :
        M[ligne, ligne - 1], M[ligne, ligne], M[ligne, ligne + 1] = \
                -r, 1. + 2. * r, -r
    M[-1, -2], M[-1, -1] = -r, 1. + 2. * r

    # Definition de v
    v = np.zeros((N, 1))

    v[0, 0], v[-1, 0] = T_int, T_ext

    # Calcul de T^1
    d = T0 + r * v
    res = CalcTkp1(M, d)
    for i in range(N):
        T_tous_k[i, 1] = res[i]

    # Calcul des T^k
    k = 2
    condition = True

    while condition and k < ItMax - 1 :
        D = np.zeros((N, 1))
        for i in range(N):
            D[i] = T_tous_k[i, k - 1] + r * v[i]
        res = CalcTkp1(M, D)
        for i in range(N):
            T_tous_k[i, k] = res[i]

        if calc_norme(T_tous_k[:, k] - T_tous_k[:, k - 1]) < epsilon :
            condition = False
        k += 1  # nbIter = k - 1

    return (k - 1, T_tous_k)
```

II.C. Programme principal

II.C.1. Début du programme

II.C.1.a. Déclaration des constantes.

```
epais = 40e-2 ; conduc = 1.65 ; rho = 2150 ; Cp = 1000
Tint = 293 ; Text1 = 283 ; Text2 = 263
epsilon = 1e-2 ; N = 60 ; Dt = 25
```

II.C.1.b. Calcul des coefficients a et b.

```
a = (Text1 - Tint) / epais
b = Tint
```

II.C.1.c. Discrétisation spatiale.

```
Dx = epais / (N + 1)
x = np.linspace(0, epais, N + 2)
```

II.C.1.d. Profil de température à l'instant initial.

```
T0 = np.zeros((N, 1))
for i in range(1, N + 1):
    T0[i - 1, 0] = a * x[i] + b
```

II.C.1.e. Calcul des coefficients du modèle thermique.

```
alpha = rho * Cp / conduc
r = Dt / (alpha * pow(Dx, 2))
```

II.C.2. Calcul des températures

II.C.2.a. Choix du schéma numérique.

```
fonctions = [schema_implicite, schema_explicite]

ind_fcts = int(input('''Choix du schema numerique :
0 : schema implicite
1 : schema explicite
'''))
```

II.C.3. Analyse du résultat

II.C.3.a. Tracé des profils de température.

```
nbIter, T = fonctions[ind_fcts](T0, ItMax, Dt, N, epais, conduc, rho, \
                                Cp, Tint, Text2, 1e-2)

plt.plot(x, [Tint] + list(T[:, 0]) + [Text1])

for i in range(100, nbIter, 100):
    # plt.plot(x, T[:, i])
    plt.plot(x, [Tint] + list(T[:, i]) + [Text2], color = 'green')

plt.plot(x, [Tint] + list(T[:, nbIter - 1]) + [Text2], color = 'red')

plt.title(str(fonctions[ind_fcts])[9:26])

plt.show()
```

Les profils obtenus sont exposés figure 2.

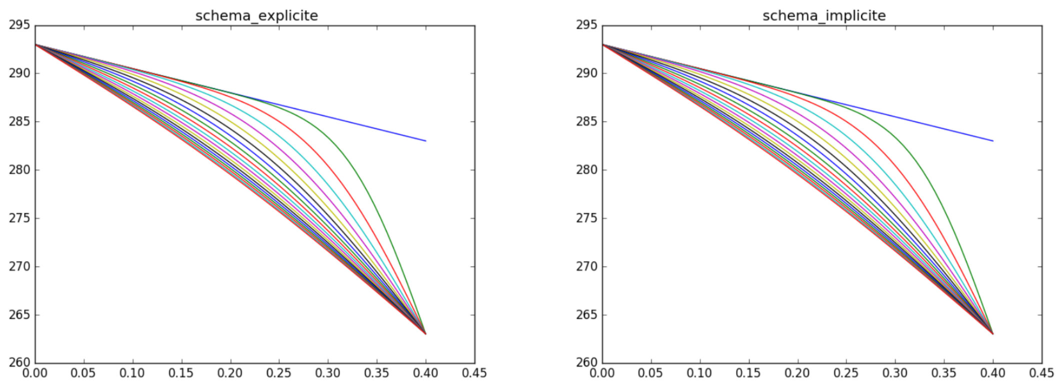

Figure 2 – Tracés des profils de température simulés

II.C.3.b. Durée du régime transitoire.

```
print('La␣durée␣au␣bout␣duquel␣le␣régime␣permanent␣est␣établi␣est␣de', \
      Dt * nbIter / 3600, 'heures')
```