Résolution numérique de l'équation de Laplace, applications en électrostatique et thermique

Mickaël Melzani
mickael.melzani@gmail.com
www.mmelzani.fr

Résumé

Cet article décrit la méthode itérative de Jacobi pour résoudre numériquement les équations de Laplace ou de Poisson, ainsi que des améliorations de cette méthode (Gauss-Seidel et sur-relaxation) qui, tout en restant simples et appropriées pour une séance de TP dans le supérieur, permettent de gagner un facteur N en complexité. Ce gain permet de simuler des situations intéressantes sur des ordinateurs modestes. Nous décrivons comment implémenter des conditions aux bords de type Dirichlet ou de type Neumann (flux fixé et cas particulier du flux nul, ou flux fixé par une loi conducto-convective). Nous présentons des exemples qui permettent pour certains de valider l'algorithme et l'implémentation des conditions aux bords par comparaison avec une solution analytique (barreau calorifugé sur les côtés avec température ou flux fixé aux extrémités, ailette de refroidissement très fine), et pour d'autres de fournir une solution numérique là où une solution analytique n'est pas disponible (condensateur, étude de l'effet de pointe, ailette de refroidissement de dimensions quelconques, pont thermique).

Introduction

Une fonction suit l'équation de Laplace si son laplacien est nul, $\Delta f = 0$. Cette équation est rencontrée dans de nombreux domaines de la physique : en électrostatique pour le potentiel dans une zone vide de charges, en conduction thermique pour la température en régime stationnaire, en diffusion moléculaire avec la densité de particules, en gravitation pour le potentiel gravitationnel dans une zone vide de masses, en hydrodynamique pour le potentiel des vitesses dans un écoulement bidimensionnel irrotationnel et incompressible, pour décrire les déformations d'une membrane, etc. On peut écrire des solutions analytiques au prix de plus ou moins d'efforts dans des situations simples, mais il est parfois plus visuel ou même nécessaire de recourir à une intégration numérique dans des cas plus généraux. Une telle résolution numérique figure par ailleurs dans les programmes de physique-chimie de certaines filières de CPGE, et peut également être proposée pour une séance de TP en licence ou être abordée dans le cadre de projets d'étudiants.

C'est dans cette optique que nous décrivons un algorithme de résolution de l'équation de Laplace. La partie 1 décrit l'algorithme et en présente certaines propriétés, et la partie 2 présente des exemples dans le cadre de l'électrostatique et de la conduction thermique en régime stationnaire. Les annexes en fin d'article présentent des compléments et donnent le code de l'algorithme et son implémentation dans le cas de l'exemple 2.4.2-iii. Tous les autres codes sont disponibles dans un fichier joint. Le langage utilisé est le langage Python (2 ou 3).

Nous avons personnellement mis en place une séance où les étudiants sont guidés pour écrire l'algorithme présenté en partie 1, puis l'appliquent sur les exemples 2.1 et 2.2. Certaines fonctions d'initialisation du domaine et de tracé graphique étaient fournies.

L'objectif de cet article n'est pas d'être exhaustif sur le sujet des méthodes numériques, ni rigoureux sur les justifications mathématiques. On pourra consulter des ouvrages spécialisés, comme le classique *Numerical Recipes* [1] qui est à la fois précis et abordable. Notons enfin qu'il existe un article à ce sujet dans le BUP [2], qui amène a constater l'augmentation de la puissance des machines depuis 1985 et ce que cela permet !

1 Algorithme de résolution

L'équation de Laplace (ou celle de Poisson $\Delta f + g = 0$ avec g fonction connue) est le prototype des équations aux dérivées partielles dites elliptiques. Ces équations servent à décrire un problème aux limites (*boundary value problem* en anglais) où les valeurs de la fonction dans le domaine doivent être déduites de celles sur les bords du domaine. Par opposition aux problèmes de Cauchy (équation de propagation, équation de la chaleur...), il n'y a pas d'évolution temporelle.

Différentes méthodes numériques sont disponibles pour la résolution de ce type d'équation : par différences finies, éléments ou volumes finis, ou par méthodes semi-analytiques utilisant les fonctions de Green ou les transformées de Fourier, ou encore les transformations conformes dans le cas bidimensionnel. Nous nous intéressons ici à une méthode par différences finies, c'est-à-dire que la fonction f inconnue est discrétisée aux points (i, j) d'un maillage (tel que celui de la figure ci-dessous). Il faut alors également discrétiser l'équation à résoudre, ce qui mène à un système d'équations portant sur les $f(x_i, y_j)$ qu'il faut résoudre. Au sein même des schémas par différences finies on trouve une variété de méthodes dont deux grandes familles : celles par écriture matricielle du système puis inversion directe de la matrice, et celles par méthode de convergence itérative (aussi dite de relaxation). C'est cette dernière qui nous intéresse ici. Il s'agit d'une méthode plutôt lente, mais qui est particulièrement simple à mettre en place et qui fournit des résultats tout à fait convenables. Il en existe évidemment des raffinements toujours plus complexes et efficaces que nous ne présentons pas (voir [1]).

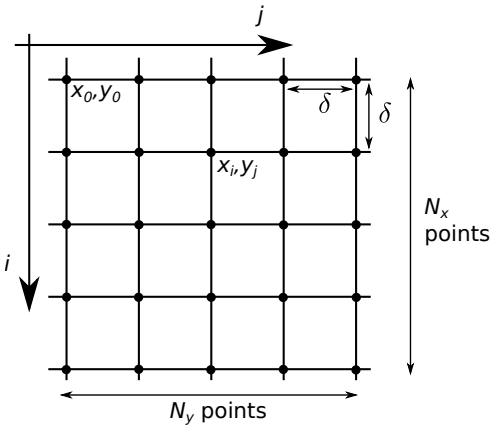
1.1 Définition de la grille et du domaine

On se restreint au cas à 2D. Le cas à 3D ne pose pas de difficultés supplémentaires, mais requiert des temps de calculs plus longs et des méthodes de visualisation des résultats plus complexes. L'espace est donc un rectangle de dimensions $L_x \times L_y$, représenté par une grille de taille $N_x \times N_y$.

La distance entre deux points de la grille est le pas spatial δ. On a ainsi $L_x = (N_x - 1)\delta$ et $L_y = (N_y - 1)\delta$.

On note (x_i, y_j) les coordonnées du point (i, j) de la grille. i va de 0 à $N_x - 1$ et j de 0 à $N_y - 1$.

Un carré de côté δ délimité par quatre points de la grille est une cellule.



La grandeur physique f (celle qui satisfait $\Delta f = 0$) est représentée par un tableau f, avec $f[i, j]$ donnant la valeur de $f(x_i, y_j)$ au point (i, j) de la grille.

Une propriété importante de l'équation de Laplace (et de celle de Poisson également) est qu'elle admet une unique solution dans un domaine $\mathcal{D}$ de l'espace si on connaît en tout point du bord de $\mathcal{D}$: soit la valeur de f (condition dite de Dirichlet), soit la valeur de la dérivée normale * de f (condition dite de Neumann) †.

Les bords du domaine peuvent simplement être le cadre de la grille, mais ils peuvent aussi inclure d'autres points de la grille pouvant représenter les armatures d'un condensateur, un conducteur métallique à un potentiel V_0 fixe, une surface à une température imposée, etc.

Nous introduisons donc le tableau B, de taille $N_x \times N_y$, tel que :

— Si $B[i, j] = 1$, alors la case (i, j) de la grille appartient au bord du domaine, et la valeur $f[i, j]$ est imposée soit par une condition de Dirichlet (sa valeur est alors toujours fixe), soit par une condition de Neumann (il faudra alors l'actualiser de façon à ce que le flux sortant du domaine à ce point soit bien celui voulu, voir partie 1.5).

— Si $B[i, j] = 0$, alors la case (i, j) n'appartient pas au bord du domaine. Il faudra alors utiliser l'équation (4), (10) ou (11) selon le schéma numérique retenu.

1.2 Discrétisation de l'équation

La fonction f étant connue uniquement aux points de la grille, il est nécessaire de discrétiser l'opérateur laplacien, ici en coordonnées cartésiennes et à deux dimensions.

Un développement limité à l'ordre 3 de f autour du point (i, j) donne :

$$f(x_i \pm \delta, y_j) = f(x_i, y_j) \pm \delta \left(\frac{\partial f}{\partial x}\right)_y (x_i, y_j) + \frac{\delta^2}{2} \left(\frac{\partial^2 f}{\partial x^2}\right)_y (x_i, y_j) \pm \frac{\delta^3}{3!} \left(\frac{\partial^3 f}{\partial x^3}\right)_y (x_i, y_j) + \mathcal{O}(\delta^4).$$

En ajoutant $f(x_i - \delta, y_j)$ et $f(x_i + \delta, y_j)$ on en déduit une écriture discrète de la dérivée seconde :

$$\left(\frac{\partial^2 f}{\partial x^2}\right)_y [i, j] = \frac{f[i+1, j] + f[i-1, j] - 2f[i, j]}{\delta^2} + \mathcal{O}(\delta^2). \tag{1}$$

∗. La dérivée normale est définie comme $\dfrac{\partial f}{\partial n} = \vec{n} \cdot \vec{\nabla} f$ si $\vec{n}$ est la normale sortante de la surface. Pour un domaine rectangulaire il s'agira simplement de $\dfrac{\partial f}{\partial x}$ ou $\dfrac{\partial f}{\partial y}$.

†. Si la condition est du type Neumann sur tous les bords, alors la solution est en fait connue à une constante additive près.

On procède de même dans la direction y, et on obtient ainsi :

$$\Delta f[i,j] = \frac{f[i+1,j] + f[i-1,j] + f[i,j+1] + f[i,j-1] - 4f[i,j]}{\delta^2} + \mathcal{O}(\delta^2). \tag{2}$$

L'équation de Laplace impose $\Delta f = 0$. Ceci peut donc s'écrire, pour tout i, j dans le domaine et à l'ordre deux en δ :

$$\boxed{f[i,j] = \frac{f[i+1,j] + f[i-1,j] + f[i,j+1] + f[i,j-1]}{4}.} \tag{3}$$

Cette expression rappelle le fait que pour une fonction harmonique (qui satisfait $\Delta f = 0$), $f(M)$ est égal à la valeur moyenne de f sur une sphère entourant M.

1.3 Méthode itérative de Jacobi

1.3.1 Description de la méthode

La méthode de Jacobi consiste à appliquer l'équation (3) pour construire par itérations les valeurs de f sur la grille, et ceci jusqu'à ce que les valeurs de f n'évoluent quasiment plus.

L'algorithme est le suivant :

- **Initialisation**
  - On crée la matrice B et on l'initialise avec des 0 ou des 1 pour décrire les bords du domaine souhaité.
  - On crée la matrice f et on l'initialise avec les valeurs voulues sur les bords du domaine (là où les conditions sont de type Dirichlet), et avec des 0 ailleurs [‡].

  Cette étape dépend donc du problème que l'on veut traiter.
- **Itérations**

  Une itération consiste à effectuer, pour tout point (i, j) qui n'appartient pas à un bord, le calcul d'une nouvelle valeur de f selon l'équation (3). Ainsi f à l'itération k est obtenue en fonction de f à l'itération $k-1$ via :

$$f_k[i,j] = \frac{f_{k-1}[i+1,j] + f_{k-1}[i-1,j] + f_{k-1}[i,j+1] + f_{k-1}[i,j-1]}{4}. \tag{4}$$

  En Python ceci nécessite de faire une copie de la matrice f, comme suit [§] :

```
fcopy = f.copy()
for i in range(Nx):
  for j in range(Ny):
    if B[i,j]==0:
      f[i,j] = (fcopy[i+1,j]+fcopy[i,j+1]+fcopy[i-1,j]+fcopy[i,j-1])/4.
```

  Attention, dans le cas de conditions aux bords de Neumann il faudra également actualiser les points appartenant aux bords correspondants, voir § 1.5.
- **Critère de terminaison**

  On stoppe la simulation lorsque les itérations ne font quasiment plus évoluer les valeurs de f. On définit pour cela l'écart

$$e = \sqrt{\frac{1}{N_x N_y} \sum_{i,j} [f_k(x_i, y_j) - f_{k-1}(x_i, y_j)]^2}, \tag{5}$$

‡. Ou avec une autre valeur que 0, qui peut être choisie pour accélérer la convergence.

§. On peut aussi utiliser

```
f[1:-1,1:-1] = (fcopy[2:,1:-1]+fcopy[1:-1,2:]+fcopy[:-2,1:-1]+fcopy[1:-1,:-2])/4.
```

sans boucle for, à condition que les bords du domaine soient restreints au cadre, ceci permettant de gagner en temps de calcul car Python optimise les additions de tableaux.

qui représente l'écart entre la nouvelle valeur de f et l'ancienne en moyenne sur toute la grille.
Une façon alternative est de calculer l'écart e comme

$$e = \max_{i,j} |f_k(x_i, y_j) - f_{k-1}(x_i, y_j)|. \tag{6}$$

On arrête le programme lorsque e devient inférieur à une valeur ε fixée à l'avance, que l'on appellera critère de convergence.

Pour un critère de convergence ε fixé, la méthode de Jacobi converge en $\mathcal{O}(N^2)$ itérations (si $N_x = N_y = N$) [1]. Comme chaque itération requiert le parcours de la grille et donc $\mathcal{O}(N^2)$ opérations, la complexité globale est en $\mathcal{O}(N^4)$.

1.3.2 Idées mathématiques derrière la méthode

Ce paragraphe a pour but de donner une intuition du pourquoi de la méthode. On donne deux interprétations :

- **(i)** Si on introduit le vecteur $\vec{f_k}$ possédant $N_x N_y$ coordonnées (les $f_k[i,j]_{i\in[0..N_x-1],j\in[0..N_y-1]}$), les $N_x N_y$ équations (4) peuvent s'écrire comme une équation de récurrence faisant intervenir une matrice M de taille $N_x N_y \times N_x N_y$:

$$\boxed{\vec{f_k} = M\vec{f}_{k-1}.} \tag{7}$$

  $\vec{f_0}$ est fixé initialement par l'utilisateur. La matrice M a des coefficients fixes, en partie donnés par l'équation (4) et en partie par les conditions aux bords utilisées.
  L'écart e s'écrit alors à l'itération k comme $e_k = \|\vec{f_k} - \vec{f}_{k-1}\|$, la norme étant au choix la norme 2 ou la norme max selon que l'on utilise la définition (5) ou (6) pour e. Le critère de convergence retenu est de fixer un ε petit et d'atteindre un k assez élevé pour que $e_k < \varepsilon$: il s'agit donc de s'assurer que $\lim_{k\to+\infty} \|\vec{f_k} - \vec{f}_{k-1}\| = 0$. Ceci va impliquer avec une bonne probabilité que la suite $\vec{f_k}$ converge [¶].
  Si on note $\vec{f}_{\text{lim}}$ la limite associée, alors on est assuré par passage à la limite que $\vec{f}_{\text{lim}} = M\vec{f}_{\text{lim}}$, c'est-à-dire que la convergence a lieu vers des valeurs $f_{\text{lim}}[i,j]$ qui vérifient l'équation discrétisée de Laplace (l'équation (3)), ce qui est bien ce que l'on souhaite.
- **(ii)** Une autre façon d'interpréter ce schéma numérique est de partir de l'équation de diffusion

$$\frac{\partial f}{\partial t} = D\Delta f. \tag{8}$$

  La solution de l'équation de Laplace est alors obtenue comme la solution une fois l'état stationnaire atteint. Si on voulait intégrer numériquement cette équation on pourrait utiliser l'expression discrète (2) pour le laplacien, et un schéma explicite d'ordre 1 en temps :

$$\frac{f_k[i,j] - f_{k-1}[i,j]}{\Delta t} = \frac{D}{\delta^2}\left(A_{i,j} - 4f_{k-1}[i,j]\right), \tag{9}$$

  avec $A_{i,j} \equiv f_{k-1}[i+1,j] + f_{k-1}[i-1,j] + f_{k-1}[i,j+1] + f_{k-1}[i,j-1]$. On peut montrer que cette méthode est stable numériquement à condition que $\dfrac{4D\Delta t}{\delta^2} \leq 1$.

¶. Mais pas à coup sûr, puisqu'il existe des suites vérifiant $\lim_{k\to+\infty}\|u_{k+1} - u_k\| = 0$ mais qui ne convergent pas. $u_k = \ln k$ par exemple. Pour savoir si la convergence a lieu il faut en connaître davantage sur la matrice M, qui dépend de l'équation que l'on résout. Il se trouve que pour l'équation de Laplace, et de façon générale pour la plupart des équations elliptiques rencontrées en physique, la matrice M a les propriétés nécessaires pour assurer la convergence [1].

Prenons donc Δt le plus large possible. On a alors $\frac{D\Delta t}{\delta^2} = \frac{1}{4}$, le terme en $f_{k-1}[i,j]$ se simplifie et l'équation (9) se met précisément sous la forme de l'équation (4). Ceci montre que la méthode itérative de Jacobi s'apparente à une convergence vers un état stationnaire via l'intégration de l'équation de diffusion (discrétisée selon (9)) et un choix de pas de temps maximal.

En particulier, la solution converge en partant de l'état initialisé au départ vers la solution du problème par diffusion à partir des bords où f (ou sa dérivée) est imposée, ce qui se voit très bien lorsque l'on trace f au fur et à mesure des itérations (voir figure 1).

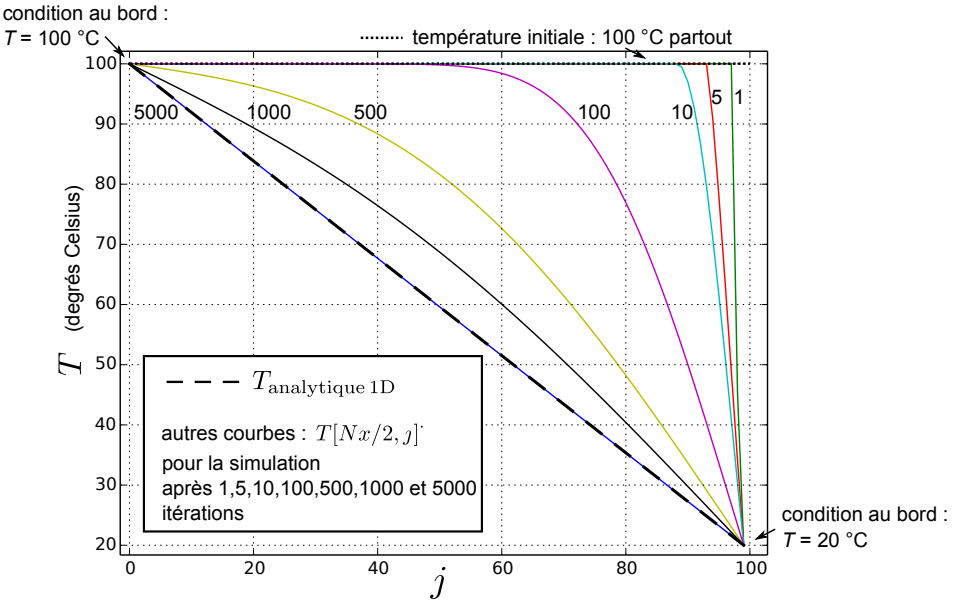


FIGURE 1 – Tracé de la solution au fur et à mesure de l'exécution du programme. Il s'agit de la température dans un barreau calorifugé sur les côtés, soumis à une température $T_0 = 100\,°C$ à son extrémité gauche, et $T_1 = 20\,°C$ à son extrémité droite (il s'agit de la simulation décrite en 2.4, cas i). On voit que la solution numérique converge vers la solution analytique en diffusant depuis le bord droit.

1.4 Amélioration par les méthodes de Gauss-Seidel et de sur-relaxation

En pratique, la complexité en $\mathcal{O}(N^4)$ (avec $N \simeq N_x \simeq N_y$) ne permet guère de dépasser des domaines de taille N de l'ordre de 100 tout en gardant des temps de simulations raisonnables pour une séance de TP. Il se trouve qu'il existe des améliorations qui permettent de faire tomber la complexité en $\mathcal{O}(N^3)$, tout en conservant un algorithme simple.

Ces améliorations sont de deux natures :

- **(i)** D'abord, on utilise la méthode de Gauss-Seidel, qui remplace l'équation (4) de Jacobi par :

$$f_k[i,j] = \frac{f_{k-1}[i+1,j] + f_k[i-1,j] + f_{k-1}[i,j+1] + f_k[i,j-1]}{4}, \tag{10}$$

où les valeurs de f en $(i-1,j)$ et $(i,j-1)$ sont celles de l'itération courante k. Ceci nécessite de parcourir la grille à i et j croissants, afin que ces valeurs aient été déjà actualisées lors de cette itération. Comme les valeurs courantes ont déjà été mises à jour, cela permet d'améliorer la vitesse de convergence, mais avec un gain d'un facteur 2 seulement [1]. En pratique dans l'algorithme il ne faudra plus utiliser une matrice copie de celle de f, mais directement itérer avec

```
for i in range(Nx):
  for j in range(Ny):
    if B[i,j]==0:
      f[i,j] = (f[i+1,j]+f[i,j+1]+f[i-1,j]+f[i,j-1])/4.
```

- **(ii)** Ensuite, on utilise une méthode dite de sur-relaxation successive, dans laquelle la nouvelle valeur est obtenue comme une moyenne pondérée entre l'estimation donnée par Gauss-Seidel et l'ancienne valeur de f au point considéré :

$$\boxed{f_k[i,j] = (1-\omega) f_{k-1}[i,j] + \omega \, \frac{f_{k-1}[i+1,j] + f_k[i-1,j] + f_{k-1}[i,j+1] + f_k[i,j-1]}{4}.} \tag{11}$$

  Le facteur de pondération ω est appelé le facteur de relaxation. La méthode est dite de sur-relaxation si $\omega > 1$ (et de sous-relaxation si $\omega < 1$, mais ce ne sera jamais notre cas car ceci ralentit la convergence).

En pratique on itérera, toujours à i et j croissants :

```
for i in range(Nx):
  for j in range(Ny):
    if B[i,j]==0:
      f[i,j] = (1.-w)*f[i,j] + w*(f[i+1,j]+f[i,j+1]+f[i-1,j]+f[i,j-1])/4.
```

On a les propriétés suivantes [1, 3] :

- L'algorithme converge si et seulement si $0 < \omega < 2$.
- Il existe une valeur optimale de ω qui permet d'atteindre le critère de convergence en $\mathcal{O}(N)$ itérations pour une grille de taille $N_x = N_y = N$:

$$\omega_{\text{opt}} \simeq \frac{2}{1 + \frac{\pi}{N}}. \tag{12}$$

  Si la grille est de taille $N_x \times N_y$, remplacer N par $N_x N_y \sqrt{2/(N_x^2 + N_y^2)}$. Ce résultat est valide pour l'équation de Laplace ou de Poisson, avec conditions aux bords de Dirichlet ou de Neumann, et est obtenue par un développement asymptotique en $N_x, N_y \gg 1$.
  Dans ce cas, la complexité totale de l'algorithme est donc en $\mathcal{O}(N^3)$ au lieu de $\mathcal{O}(N^4)$, ce qui est un gain significatif et même nécessaire pour réaliser des simulations intéressantes. C'est avec cette version que nous travaillerons dans toute la suite.
- Pour $\omega = 1$ on retrouve la méthode de Gauss-Seidel.
- Cette méthode nécessite un ordre particulier de parcours de i et j pour fonctionner. Le sens de parcours de la méthode de Gauss-Seidel fonctionne, mais d'autres possibilités existent.

Enfin, notons que d'autres méthodes d'optimisation sont possibles (accélération de Chebyshev qui change la valeur de ω au cours des itérations, schéma multigrilles qui utilise des grilles de plus en plus fines...), mais sont significativement plus complexes à mettre en place. Voir par exemple [1, 3].

1.5 Traiter des conditions aux bords de Neumann

Les conditions de type Dirichlet ne posent aucun problème particulier : il suffit de ne pas actualiser les valeurs de f là où elles sont fixées. Dans les extraits de code ci-dessus, ceci est assuré par le test `if B[i,j]==0` qui permet d'exclure les bords du domaine. Les conditions de type Neumann nécessitent en revanche d'être détaillées, c'est ce que nous faisons dans cette partie. Nous donnons ensuite des exemples concrets dans le cas du transfert thermique dans la partie 2.

1.5.1 Principe

On suppose par exemple que le flux est imposé sur le bord droit du domaine, donc pour les cases $[i, N_y - 1]$ avec i allant de 1 à $N_x - 2$ [‖]. On note $\varphi(i)$ le flux (qui est fixé) sur la case

‖. On exclut les coins ($[0, N_y - 1]$ et $[N_x - 1, N_y - 1]$) car ceux-ci ne servent jamais dans les équations d'itération. On peut toutefois les actualiser à la valeur moyenne des cases autour pour des critères esthétiques sur les représentations graphiques.

$[i, N_y - 1]$:

$$\frac{\partial f}{\partial x}(x_i, y_{N_y-1}) = \varphi(i).$$

Il faut évaluer de façon discrète la dérivée première. La façon la plus simple est le schéma décentré d'ordre 1 ** :

$$\frac{\partial f}{\partial x}(x_i, y_{N_y-1}) = \frac{f[i, N_y - 1] - f[i, N_y - 2]}{\delta} + \mathcal{O}(\delta),$$

ce qui donne alors à l'ordre 1 en δ :

$$\boxed{f[i, N_y - 1] = f[i, N_y - 2] + \delta \times \varphi(i).} \tag{13}$$

Il suffit donc, lors du parcours à i croissant puis j croissant de toute la grille, d'appliquer l'équation (13) pour actualiser le point dès que $j = N_y - 1$. On a donc par exemple :

```
for i in range(Nx):
  for j in range(Ny):
    if B[i,j]==0:
      f[i,j] = (1.-w)*f[i,j] + w*(f[i+1,j]+f[i,j+1]+f[i-1,j]+f[i,j-1])/4.
    if j==Ny-1 and 0<i<Nx-1:
      f[i,j] = f[i,j-1] + delta*phi[i]
```

On procède évidemment de même si on veut implémenter ce type de conditions sur les autres bords.

Dans le cas particulier (et important) d'une paroi calorifugée le flux doit être nul, donc $\varphi(i) = 0$ et on a simplement

$$\boxed{f[i, N_y - 1] = f[i, N_y - 2].} \tag{14}$$

1.5.2 Application : flux conducto-convectif via la loi de Newton

La loi de Newton est souvent utilisée pour modéliser simplement des échanges conducto-convectifs entre un solide et un fluide. Cette loi impose un flux $\varphi(i) = k\{f(x_i, y_{N_y-1}) - f_{\text{ext}}\}$, avec k une constante de proportionnalité et f_{ext} une constante (dans le cas thermique on a $k = -h/\lambda$ et $f_{\text{ext}} = T_{\text{ext}}$, voir partie 2 pour des illustrations).

On considère le même bord que plus haut. On a donc, avec le même schéma décentré d'ordre 1 :

$$\frac{f[i, N_y - 1] - f[i, N_y - 2]}{\delta} = k(f[i, N_y - 1] - f_{\text{ext}}),$$

d'où

$$\boxed{f[i, N_y - 1] = \frac{f[i, N_y - 2] + a f_{\text{ext}}}{1 + a}, \quad \text{avec} \quad a = -\delta k.} \tag{15}$$

**. On peut noter que l'évaluation de la dérivée est d'ordre 1 en δ, ce qui fait chuter l'ordre global du schéma numérique. Il est possible d'implémenter des schémas qui restent d'ordre 2 en δ, en utilisant une évaluation centrée de la dérivée. Par exemple $\frac{f[i, N_y] - f[i, N_y - 2]}{2\delta} = \frac{\partial f}{\partial y}(x_i, y_{N_y-1}) + \mathcal{O}(\delta^2)$, donc $\frac{f(i, N_y) - f(i, N_y - 2)}{2\delta} = \varphi(i)$. Ceci fait intervenir un point fictif en $j = N_y$ qui est hors du domaine. L'idée est alors d'utiliser l'équation (4) appliquée en $(i, N_y - 1)$ pour obtenir une expression de f au point fictif. Cependant ceci ne semble pas simple dans le cas de l'utilisation des méthodes de Gauss-Seidel et de sur-relaxation, car $f[i, N_y - 1]$ n'est pas donnée par (4) mais par (11), qui fait intervenir des valeurs à des temps différents, ce qui complique les choses. Nous n'avons pas poussé plus loin dans cette direction, les exemples présentés ici montrant que le schéma d'ordre 1 est satisfaisant.

2 Exemples d'applications en électrostatique et thermique

Cette partie présente des exemples dans deux domaines où l'équation de Laplace est rencontrée : en électrostatique dans un domaine vide de charges le potentiel électrostatique suit l'équation $\Delta V = 0$, et en conduction thermique où en régime stationnaire l'équation de la chaleur sans source dans un solide homogène s'écrit $\Delta T = 0$.

Le tableau ci-dessous récapitule les différents exemples.

Exemple	Domaine	Objectifs	Conditions aux bords
2.1 - Condensateur dans une boîte	électrostatique	Application simple et visuelle	Dirichlet
2.2 - Étude de l'effet de pointe	électrostatique	Idem au-dessus, avec en plus une mesure de l'amplification du champ électrique et une discussion de l'influence de ε et N	Dirichlet
2.3 - Solide avec T fixée sur les quatre faces	thermique	Simple et visuel	Dirichlet
2.4 - Barre calorifugée sur les côtés	thermique	Permet une validation de l'algorithme et des conditions aux limites par comparaison avec la solution analytique	Dirichlet, Neumann (flux fixé, nul ou loi de Newton)
2.5 - Ailette de refroidissement	thermique	Idem au-dessus, et peut s'étendre à des cas où la résolution analytique 1D n'est plus valide	Dirichlet, Neumann (flux nul ou loi de Newton)
2.6 - Pont thermique dans un bâtiment	thermique	Cas où une solution analytique n'est pas disponible. Situation pleinement 2D.	Idem au-dessus, et domaine non rectangulaire

2.1 Condensateur dans une boîte

Un exemple très simple mais illustratif est l'obtention des équipotentielles pour un condensateur plan. Le condensateur est défini par une armature au potentiel $-V_0$, et une au potentiel $+V_0$ (conditions de Dirichlet). Il faut donc initialiser la matrice B des bords en conséquence ($B[i, j] = 1$ sur les bords du cadre et à l'emplacement des armatures), et initialiser f avec les valeurs voulues sur le cadre et à l'emplacement des armatures.

On impose un potentiel nul sur le cadre du domaine, ce qui signifie que le condensateur est en réalité dans une boîte dont le bord est au potentiel nul. Si la boîte est assez grande, comme le potentiel du condensateur tend vers 0 à l'infini, celle-ci n'aura qu'une influence négligeable.

Notons que le programme étant 2D, on simule en fait ce qu'il se passe pour un condensateur infini dans la direction perpendiculaire au plan de la simulation.

On montre un exemple figure 2. Les bords et les équipotentielles sont affichés à l'aide des fonctions Python décrites dans l'annexe.

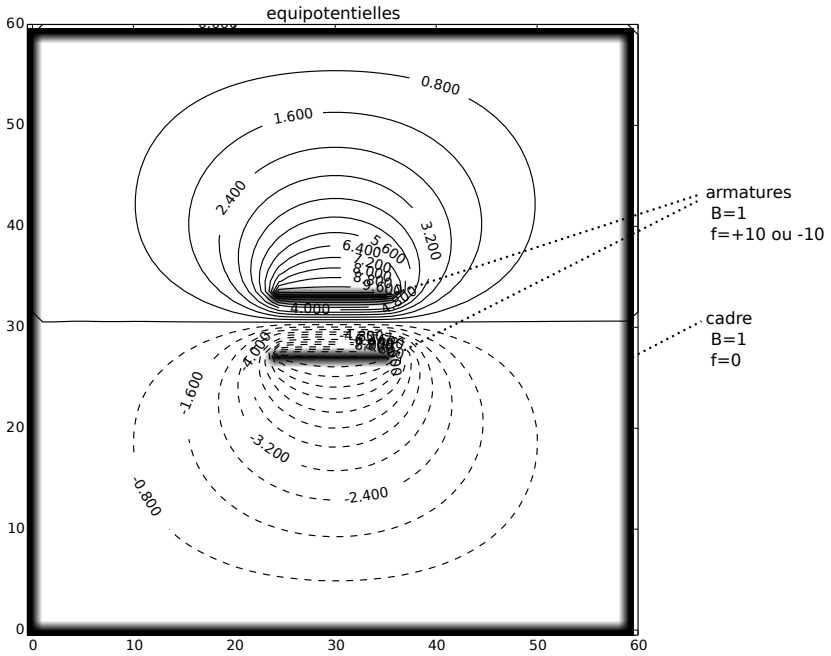


FIGURE 2 – Exemple de simulation : équipotentielles pour un condensateur.

2.2 Étude de l'effet de pointe

On présente un second exemple en électrostatique : l'étude de l'amplification du champ électrique atmosphérique par la présence d'un conducteur haut et pointu. Cet exemple a l'avantage de se rattacher à une situation concrète et de fournir des résultats chiffrés.

En l'absence de perturbations, il y a dans l'atmosphère terrestre un gradient de potentiel entre le sol (qui est à un potentiel V_{sol} que l'on prendra égal à 0 V par convention) et l'ionosphère terrestre (qui est à un potentiel de l'ordre de 3×10^5 V, voir [4]). Ainsi, entre le sol et une hauteur L donnée, on passe quasi linéairement du potentiel $V_{\text{sol}} = 0\,\text{V}$ à un potentiel $V(L)$ plus élevé. Dans une atmosphère non perturbée, par temps calme, le gradient de potentiel est de 100 V/m environ, ce qui est donc aussi la norme du champ électrique $\vec{E}$.

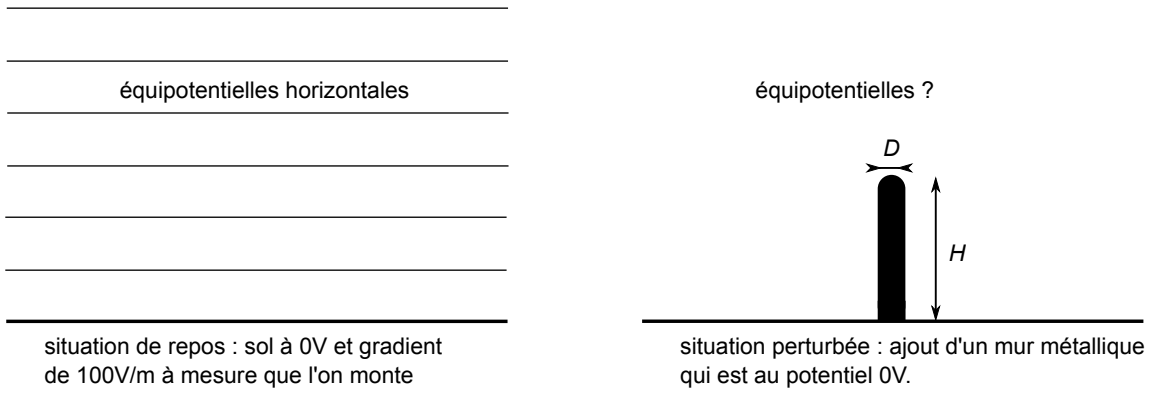


FIGURE 3 – Description du problème de l'effet de pointe.

On perturbe maintenant cette situation en plaçant un mur vertical de hauteur H et de largeur D (voir figure 3), dont le sommet est un demi-cylindre de diamètre D, fait dans un matériau conducteur et qui est donc au même potentiel en tout point ††. Ce mur est relié au sol, son potentiel est donc aussi celui du sol. Ceci va déformer les surfaces équipotentielles de l'atmosphère, et donc en particulier modifier la valeur du champ électrique. On s'attend à ce que

††. Pour simuler une tige et non pas un mur, il faudrait faire une simulation 3D.

ce dernier augmente vers le sommet : c'est ce que l'on nomme l'effet de pointe, et qui explique en partie que la foudre tombe préférentiellement sur les objets pointus.

Notre programme va permettre de simuler cette situation, et d'obtenir de façon précise les surfaces équipotentielles et surtout la valeur maximale du champ électrique en fonction de H et de D.

On initialise avec une fonction `initialisation_effet_pointe(B,V,a,b,h,d)` qui permet d'établir un domaine avec : (i) un cadre dont le bord bas est au potentiel a et le bord haut au potentiel b ; (ii) sur les bords droit et gauche du cadre, le potentiel est aussi fixé et augmente linéairement de a à b ; (iii) un mur de hauteur h et épaisseur d (en nombre de points), au potentiel fixé a. Le sommet du mur est arrondi. Un exemple de code est donné dans le fichier Python.

Dans ce qui suit, on prend un pas de grille $\delta = 3 \times 10^{-2}\,\mathrm{m}$, et :

— $h = 50$ et $d = 13$, ce qui correspond à un mur de hauteur $H = h \times \delta = 1.5\,\mathrm{m}$ et de largeur $D = d \times \delta = 39\,\mathrm{cm}$,

— un potentiel $a = 0\,\mathrm{V}$ au sol et $b = (N_x - 1)\delta \times 100\,\mathrm{V}$ en haut du cadre, ce qui donne bien un gradient au repos de $100\,\mathrm{V/m}$,

— un critère de convergence $\varepsilon = 10^{-3}\,\mathrm{V}$ défini par l'équation (5).

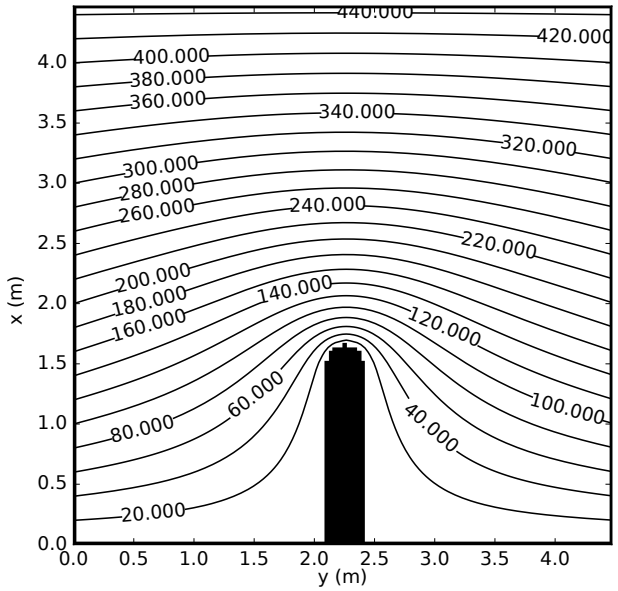


FIGURE 4 – Exemple de simulation de l'effet de pointe. On a tracé les équipotentielles (en V).

Un exemple est montré figure 4 pour un domaine de taille $N_x \times N_y = 150 \times 150$, ce qui correspond à $N_x\delta \times N_y\delta = 4.5\,\mathrm{m} \times 4.5\,\mathrm{m}$. On peut ensuite calculer la valeur de $\vec{E}$ (voir annexe 1), et extraire la valeur maximale $\|\vec{E}\|_{\max}$ de sa norme. Dans le cas présent on obtient $\|\vec{E}\|_{\max} = 508\,\mathrm{V/m}$, soit cinq fois la valeur du champ au repos.

Enfin, cette étude de l'effet de pointe permet de discuter deux points centraux en simulations numériques : la convergence des solutions (ici par rapport à la valeur de ε), et l'influence de la taille du domaine et donc des conditions aux limites.

Influence du critère de convergence ε

Dans l'idéal, on souhaiterait prendre un critère ε le plus petit possible, mais cela impliquerait des temps de calculs immenses, et il faut donc trouver un compromis entre rapidité et précision. Le critère est que ε est assez petit si les résultats de la simulation n'en dépendent quasiment plus. S'ils en dépendent notablement, c'est que ε est encore trop grand.

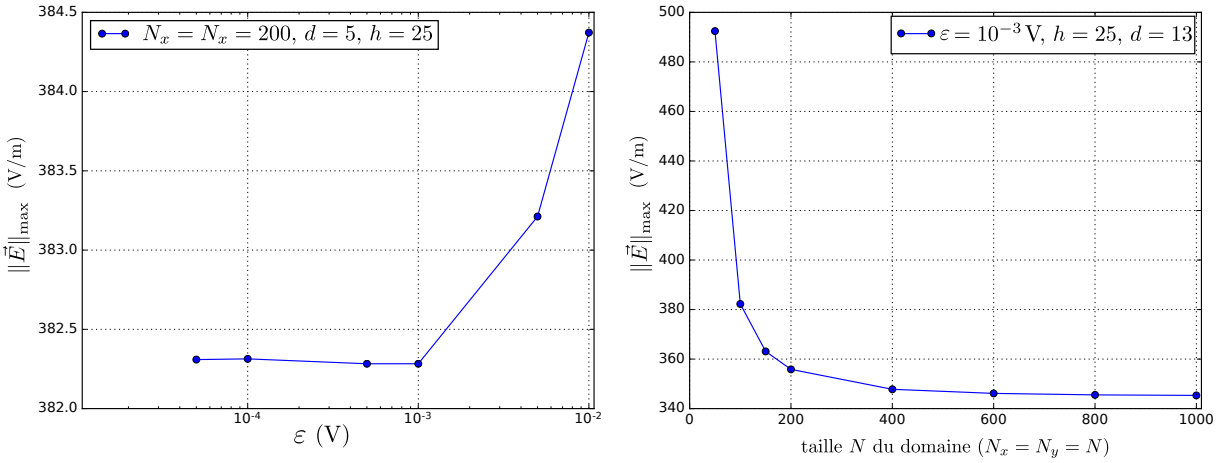


FIGURE 5 – Gauche : influence du choix du critère de convergence ε sur le résultat $\|\vec{E}\|_{\max}$ de la simulation de l'effet de pointe. Tous les paramètres sont fixés, seul ε varie. Droite : idem, mais influence de la taille du domaine pour un domaine carré. Tous les paramètres sont fixés, seul N varie. Dans chacun des deux cas, le paramètre (ε ou N) atteint une valeur suffisante lorsque le résultat n'en dépend quasiment plus.

Le graphique figure 5 à gauche donne le résultat de plusieurs simulations, pour $d = 5$, $h = 25$ et $N_x = N_y = 200$ et différentes valeurs de ε. On voit que dans ce cas la valeur de $\varepsilon = 10^{-3}\,\mathrm{V}$ convient. Il n'est pas nécessaire de prendre plus petit car cela demande plus de temps de calcul pour un résultat final identique.

Influence de la taille du domaine

Dans un cas comme celui-ci, on ne veut pas que la présence des bords haut, droit et gauche ait une influence sur la forme des équipotentielles et donc des résultats. On prendrait dans l'idéal un domaine de taille immense, mais cela implique une place en mémoire et un temps de calcul beaucoup trop grands. Il faut donc ici aussi vérifier que les résultats de la simulation ne dépendent pas de la taille du domaine.

Le graphique figure 5 à droite donne le résultat de plusieurs simulations, pour $d = 13$, $h = 25$, $\varepsilon = 10^{-3}\,\mathrm{V}$ et différentes valeurs de N ($N_x = N_y = N$). On constate qu'il faut au moins une taille de 400×400 pour avoir un résultat suffisamment indépendant de la taille du domaine. Cette valeur est difficilement atteignable en séance de TP, mais ce type de graphique permet de sensibiliser les étudiants à cette question et de leur montrer la démarche à suivre.

Remarquons enfin qu'il n'est pas possible de comparer ces valeurs à des expressions analytiques, car les résultats classiques en théorie du potentiel qui prédisent un champ $\|\vec{E}\|$ en $D^{-1/2}$ à 2D [5, §2.11] ne s'appliquent pas à cause de la présence du sol et du gradient de potentiel au repos.

2.3 Solide dont les températures sont fixées sur les quatre faces

Nous passons maintenant à des cas de conduction thermique. Le cas d'un solide dont la température est imposée sur chacune des faces est un problème avec conditions aux bords de Dirichlet uniquement. Il ne pose aucune difficulté particulière.

L'exemple figure 6 montre le champ de température dans un solide maintenu à 20 °C sur la face du haut et à 60 °C sur les trois autres faces, avec une grille de 60×80 points. Seule l'équation (11) est utilisée, il n'y a donc aucune longueur caractéristique et le pas δ n'intervient pas.

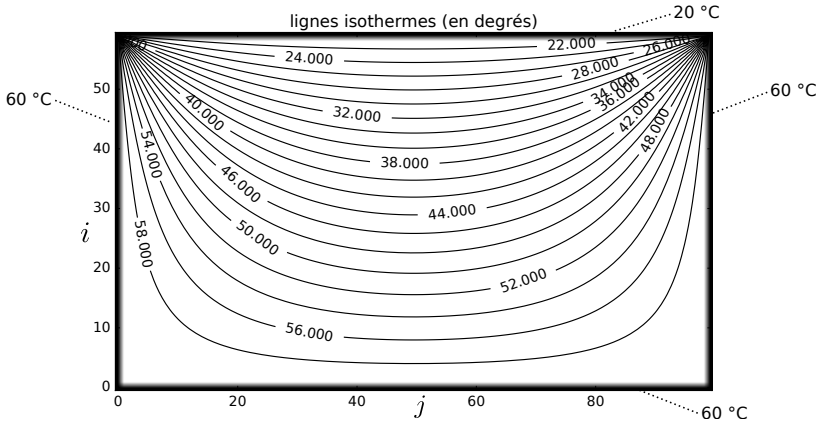


FIGURE 6 – Lignes isothermes pour un solide soumis à des conditions de Dirichlet sur ses quatre faces : 20 °C en haut et 60 °C sur les trois autres faces.

2.4 Barre calorifugée sur les côtés, conditions aux limites de Dirichlet ou de Neumann

Les exemples précédents n'ont pas permis de tester la validité de l'algorithme, ni d'utiliser les conditions aux bords de Neumann. L'objectif est donc maintenant de se placer dans une situation où tous les types de bords sont utilisés, et où une comparaison avec une solution analytique est possible. On considère pour cela un barreau solide de longueur L_y, calorifugé sur ses faces latérales (voir figure 7). En $y = 0$ le barreau est au contact d'un thermostat à la température T_0, ce qui amène à prendre une condition de type Dirichlet : $T(0) = T_0$. En $y = L_y$ on considèrera trois types de conditions aux limites :

(i) une température T_1 fixée,

(ii) un flux j_1 fixé,

(iii) un flux j_1 fixé par une relation de type conducto-convectif.

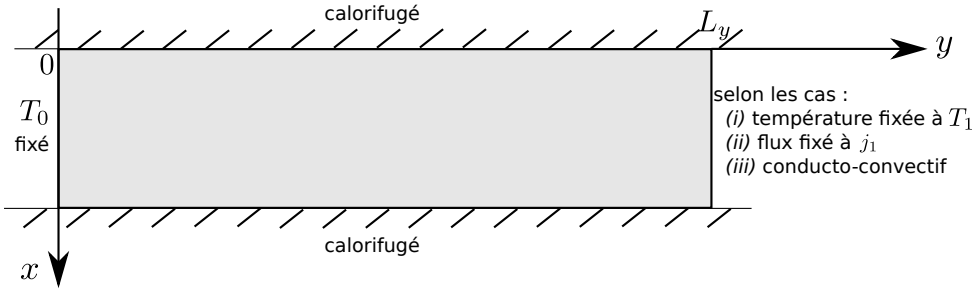


FIGURE 7 – Description du problème de la barre calorifugée sur les côtés.

2.4.1 Modèle 1D, résolution analytique

On suppose que la situation peut être décrite de manière unidimensionnelle : on a $T = T(y)$ et un vecteur densité de courant thermique $\vec{j}_{\text{th}} = j_{\text{th}}(y)\vec{e}_y$.

L'équation de la chaleur devient simplement $T''(y) = 0$, donc $T(y) = Ay + B$. Il reste à déterminer les constantes A et B avec les conditions aux limites. On a selon chacun des trois

cas :

$$\text{(i)} \qquad T(0) = T_0 \ \text{ et } \ T(L_y) = T_1, \qquad \text{d'où } \ T(y) = \frac{T_1 - T_0}{L_y}\, y + T_0. \tag{16}$$

$$\text{(ii)} \qquad T(0) = T_0 \ \text{ et } \ -\lambda T'(L_y) = j_1, \qquad \text{d'où } \ T(y) = -\frac{j_1}{\lambda}\, y + T_0. \tag{17}$$

$$\text{(iii)} \qquad T(0) = T_0 \ \text{ et } \ -\lambda T'(L_y) = h(T(L_y) - T_{\text{ext}}), \qquad \text{d'où } \ T(y) = \frac{T_{\text{ext}} - T_0}{L_y + \lambda/h}\, y + T_0. \tag{18}$$

2.4.2 Modèle 2D, résolution numérique

Les conditions sur les bords sont implémentées avec les équations (13) pour un flux fixé, (14) pour un bord calorifugé, et (15) pour une condition de type Newton.

Cas (i) – T fixe à gauche et à droite

On montre figure 8 un exemple avec $N_x \times N_y = 10 \times 100$ et $\delta = 1\,\text{cm}$ (soit une barre de 10 cm par 1 m), $T_0 = 100\,°\text{C}$, $T_1 = 20\,°\text{C}$. Le graphique correspond à $\varepsilon = 10^{-7}\,°\text{C}$ calculé comme l'écart maximal des $|T_k[i,j] - T_{k-1}[i,j]|$.

Quelques remarques :

— L'accord entre solution analytique (équation 16) et numérique est très bon.

— Il a fallu environ 7 000 itérations pour atteindre $e \leq \varepsilon$.

— On a $\varepsilon = 10^{-7}\,°\text{C}$, et l'écart maximal entre solution analytique et numérique est de $5.6 \times 10^{-5}\,°\text{C}$ (figure 8(b)). Cet écart est proportionnel à ε (du moins pour des tests avec $\varepsilon \in [10^{-10}\,°\text{C}, 10^{-5}\,°\text{C}]$).

Cet écart maximal est au milieu de la barre car le profil de température se "propage" par diffusion depuis les bords fixes en $y = 0$ et $y = L_y$ vers le centre.

— Le profil dans une tranche à y constant n'est pas tout à fait uniforme : on constate figure 8(d) une variation de $\delta_x T \simeq 4.5 \times 10^{-7}\,°\text{C}$ (et des tests pour d'autres valeurs de ε montrent que $\delta_x T = 4.5 \times \varepsilon$).

Ce profil non plat est inhérent à la méthode de parcours de la grille de Gauss-Seidel : par exemple la ligne $i = 0$ est actualisée avec l'instruction $T[0,j] = T[1,j]$, puis la ligne 1 selon l'équation (11), et donc la condition de flux nul $T[0,j] = T[1,j]$ n'est jamais respectée à la fin d'une itération. En revanche on termine le parcours par la ligne $i = N_x - 1$, donc la condition de flux nul $T[N_x - 2, j] = T[N_x - 1, j]$ est toujours respectée en fin d'itération, ce qui est bien le cas sur le graphique.

On peut supprimer cette asymétrie en utilisant la méthode de Jacobi. On obtient alors un profil en U symétrique par rapport au centre, mais toujours avec un flux non nul sur les bords car on impose $T_k[0,j] = T_{k-1}[1,j]$ et non pas $T_k[0,j] = T_k[1,j]$. De plus, le temps de convergence s'envole.

En réalité ce profil latéral est sans effet tant qu'il induit un flux thermique latéral négligeable devant le flux thermique selon y. Cela va donc dépendre de la valeur de ε et de la taille de la grille. Ici on a $\delta_x T / N_x \simeq 4.5\varepsilon / N_x = 4.5 \times 10^{-8}\,°\text{C/cellule}$ et $\delta_y T / N_y \simeq 80\,°\text{C}/100\,\text{cellules} = 8 \times 10^{-1}\,°\text{C/cellule}$, ce qui permet de dire que nos bords calorifugés jouent leur rôle de façon très convenable.

Cas (ii) – T fixe à gauche, flux fixe à droite

Même exemple que dans le (i), mais cette fois on impose à droite un flux $j_1 = 1200\,\text{W} \cdot \text{m}^{-2}$. Il faut alors préciser la conductivité, nous prenons $\lambda = 400\,\text{W} \cdot \text{K}^{-1} \cdot \text{m}^{-1}$ (valeur typique pour un métal).

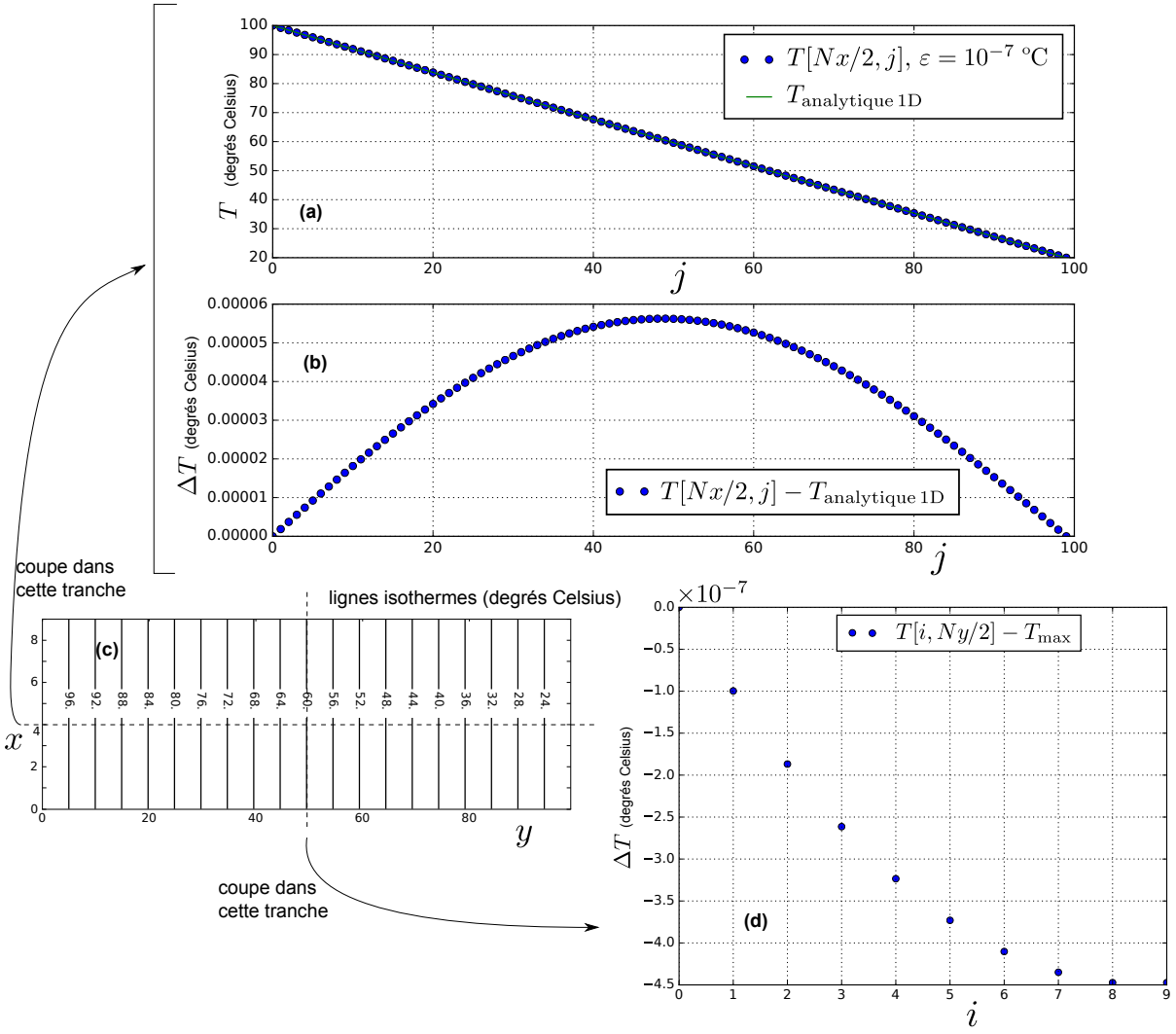


FIGURE 8 – Résultats du cas (i) : barre calorifugée sur les côtés, T fixe aux deux extrémités.

Notons que cette fois la valeur du pas δ intervient via l'équation (13). Nous ne reproduisons pas les graphiques car ils ressemblent en tout point à ceux du cas (i).

Quelques remarques :

— L'accord entre solution analytique et numérique est très bon.

— Il a fallu environ 20 000 itérations pour $\varepsilon = 10^{-7}$ °C.

— On a $\varepsilon = 10^{-7}$ °C, et l'écart maximal entre solution analytique et numérique est de 2×10^{-4} °C. Cet écart est proportionnel à ε (du moins pour des tests avec $\varepsilon \in [10^{-10}\,°\text{C}, 10^{-5}\,°\text{C}]$). Cet écart maximal est au bout de la barre car le profil de température se "propage" par diffusion depuis le bord où il est fixe en $y = 0$ vers le bord ouvert en $y = L_y$.

— Le profil dans une tranche à y constant n'est pas tout à fait uniforme : même remarque que pour le cas (i) et même conclusion quant à l'efficacité satisfaisante des bords calorifugés.

Cas (iii) – T fixe à gauche, flux conducto-convectif à droite

Même exemple que dans le (i) et (ii), mais cette fois on impose à droite un flux via une loi de type Newton, avec un coefficient conducto-convectif $h = 15\,\text{W} \cdot \text{m}^{-2} \cdot \text{K}^{-1}$ (typique pour un contact solide-air).

Pour l'évaluation de $T[i, N_y - 1]$ on utilise l'équation (15), avec $a = \dfrac{h\delta}{\lambda}$ et $f_{\text{ext}} = T_{\text{ext}}$, que

l'on prend égale à 10 °C ici. Nous ne reproduisons pas les graphiques car ils ressemblent en tout point à ceux du cas (i). Quant aux remarques, celles du cas (ii) s'appliquent également ici mot pour mot.

Conclusion

En conclusion, les résultats cette partie 2.4 permettent de valider l'algorithme et l'implémentation des conditions aux bords. L'exemple suivant s'attache à faire de même, mais dans une situation moins simple où le barreau n'est plus calorifugé sur les côtés.

2.5 Ailette de refroidissement

L'exercice de l'ailette de refroidissement est un classique souvent abordé dans le cours de conduction thermique. C'est pourquoi il peut être intéressant de traiter cette situation numériquement, et de comparer les résultats du modèle analytique 1D traité en exercice et du modèle numérique 2D. La partie 2.5.1 expose rapidement le modèle 1D habituel, et démontre sous quelles conditions une ailette 2D peut être modélisée de façon 1D. La partie 2.5.2 présente les simulations numériques et discute de leur accord avec les solutions analytiques 1D. On conclut partie 2.5.3 sur l'accord et les possibilités offertes.

On considère donc une ailette de refroidissement de section rectangulaire $L_z \times L_x$, et de longueur totale L_y. Elle est faite dans un matériau de conductivité thermique λ, et elle est en contact avec une source chaude à la température T_0 en $y = 0$, et avec de l'air à T_{ext} partout ailleurs. Voir figure 9.

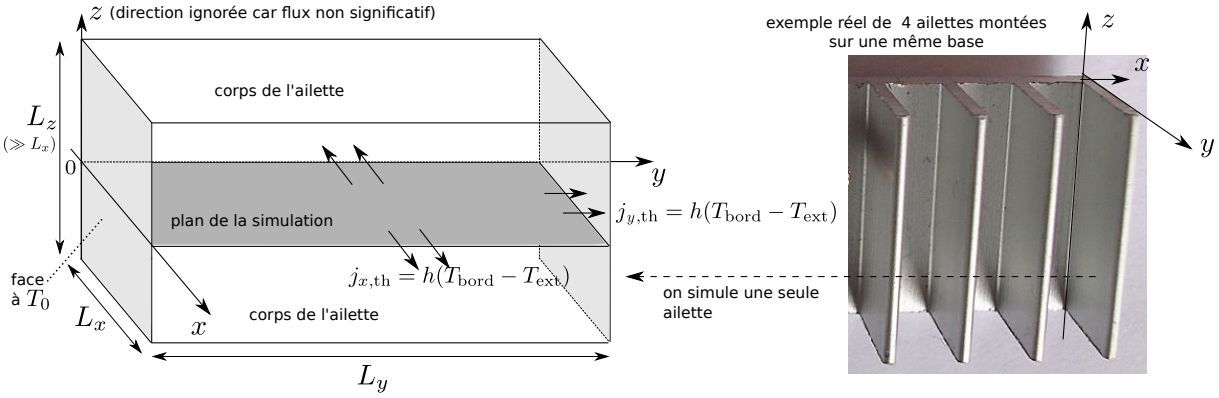


FIGURE 9 – Description du problème de l'ailette.

Les hypothèses communes à la modélisation analytique 1D et numérique 2D sont les suivantes :

— L'ailette est très large : $L_z \gg L_x$, si bien que les transferts thermiques vont avoir lieu majoritairement selon les faces parallèles à yz, et très peu à travers celles parallèles à xy. Ceci revient à négliger les transferts thermiques ayant lieu dans la direction z, ou encore à supposer que le vecteur densité de courant thermique $\vec{j}_{\text{th}}$ est dans le plan xy et que $T = T(x, y)$.

Cette hypothèse est nécessaire pour permettre des simulations numériques 2D. Nous verrons par ailleurs que pour permettre une modélisation 1D analytique, il faudra encore une condition supplémentaire.

— La condition aux limites en $y = 0$ est $T(x, 0) = T_0$ (de type Dirichlet).

La condition aux limites sur les autres faces est conducto-convective, donnée par la loi de Newton : le flux surfacique sortant par les faces parallèles à yz est $\varphi(y) = h(T_{\text{bord}}(y) - T_{\text{ext}})$

avec h coefficient conducto-convectif pris uniforme, et $T_{\text{bord}}(y)$ la température à l'abscisse y sur le bord de l'ailette (en $x = 0$ ou $x = L_x$).

2.5.1 Modèle 1D, résolution analytique

Le modèle utilisé est basé sur les hypothèses communes ci-dessus, et sur l'hypothèse supplémentaire suivante :

— On se place dans une situation 1D où la température dans une tranche $y = \text{cst}$ est quasi uniforme : $T(x, y) \simeq T(y)$, où $T(y)$ représente la valeur moyenne de la température dans la tranche.

Les transferts thermiques deviennent alors de deux types :

(i) le transfert latéral (selon $\pm\vec{e}_x$) est conducto-convectif, donné par $\varphi = h(T(y) - T_{\text{ext}})$;

(ii) le transfert longitudinal au sein de l'ailette est donné par $\vec{j}_{\text{th}} = j_{\text{th}}(y)\vec{e}_y$.

Nous discutons des conditions de validité de cette hypothèse 1D dans la suite.

La résolution analytique est alors classique et nous passons les détails. Un bilan de flux thermique sur un volume de longueur dy donne

$$j_{\text{th}}(y)L_xL_z = j_{\text{th}}(y + \mathrm{d}y)L_xL_z + 2 \times \mathrm{d}yL_zh(T(y) - T_{\text{ext}}).$$

On en déduit une équation faisant intervenir $j'_{\text{th}}(y)$, puis en utilisant la loi de Fourier on obtient :

$$\boxed{T''(y) - \frac{1}{\delta_p^2}T(y) = -\frac{1}{\delta_p^2}T_{\text{ext}}, \quad \text{avec} \quad \delta_p = \sqrt{\frac{\lambda L_x}{2h}}.} \tag{19}$$

δ_p est appelé profondeur de peau de l'ailette. La forme générale des solutions est

$$\frac{T(y) - T_{\text{ext}}}{T_0 - T_{\text{ext}}} = A_1\mathrm{e}^{-y/\delta_p} + B_1\mathrm{e}^{+y/\delta_p}. \tag{20}$$

Les constantes A_1 et B_1 sont obtenues en imposant $T(0) = T_0$ et $-\lambda T'(L_y) = h(T(L_y) - T_{\text{ext}})$, ce qui donne :

$$A_1 = \frac{1}{1 + \frac{\alpha-1}{\alpha+1}\exp\{-2\beta\}} \quad \text{et} \quad B_1 = \frac{1}{1 + \frac{\alpha+1}{\alpha-1}\exp\{+2\beta\}}. \tag{21}$$

Les constantes intervenant dans ces expressions, ainsi que leurs interprétations, sont les suivantes :

- $\beta = \dfrac{L_y}{\delta_p} = L_y\sqrt{\dfrac{2h}{\lambda L_x}}$ contrôle le caractère "infini" de l'ailette. Si $\beta \gg 1$, on peut considérer l'ailette infinie. Ceci parce qu'au bout de quelques δ_p la température est quasi égale à T_{ext}, le flux vers l'extérieur est donc quasi nul dans cette zone, et la partie de l'ailette au-delà de quelques δ_p ne joue plus aucun rôle.

  Remarquons toutefois qu'au bout de la barre le terme en B_1 contribue toujours de façon significative sur l'allure détaillée de la courbe $T(y)$, quelle que soit la longueur L_y de la barre. Pour voir ceci, prenons une ailette pour laquelle $\exp\{2\beta\} \gg 1$. En $y = L_y$ on a $A_1\mathrm{e}^{-y/\delta_p} \simeq \mathrm{e}^{-L_y/\delta_p} = \mathrm{e}^{-\beta}$, et $B_1\mathrm{e}^{y/\delta_p} \simeq \frac{\alpha-1}{\alpha+1}\mathrm{e}^{-2\beta}\mathrm{e}^{L_y/\delta_p} = \frac{\alpha-1}{\alpha+1}\mathrm{e}^{-\beta}$. Les deux termes sont très petits mais contribuent avec le même poids et influent pareillement sur l'allure de la courbe. Redisons toutefois que si $L_y \gg \delta_p$, ce terme en B_1 reste bien négligeable partout ailleurs, et le flux thermique au-delà de quelques δ_p est bien négligeable devant celui déjà évacué par les côtés.

- $\alpha = \dfrac{\lambda}{\delta_p h} = \sqrt{\dfrac{2\lambda}{hL_x}}$ contrôle le caractère "1D" de l'ailette. Si $\alpha^2 \gg 1$ alors le modèle 1D utilisé ici s'applique.

Pour démontrer ceci, notons $T_{\text{bord}}(y)$ la température du bord de l'ailette à l'abscisse y (donc en $(x = 0, y)$ ou en $(x = L_x, y)$). C'est celle-ci qui doit intervenir dans la loi de Newton : $j_{x,\text{th}} = h(T_{\text{bord}}(y) - T_{\text{ext}})$ par continuité du courant thermique selon x à l'interface air-solide. Pour aboutir à une modélisation 1D on suppose que, à y fixé, la température $T(x, y)$ ne varie pas trop avec x et est bien représentée par sa valeur moyenne $T(y)$. Notons $\delta_x T = |T(L_x/2, y) - T_{\text{bord}}(y)|$ la variation de température dans une section $y = \text{cst}$ donnée. Le critère est donc $\delta_x T$ petit, typiquement devant la différence $T_0 - T_{\text{ext}}$.

On peut évaluer $\delta_x T$ en utilisant la loi de Fourier : $\lambda \dfrac{\delta_x T}{L_x/2} \simeq j_{x,\text{th}}$, puis en écrivant que $j_{x,\text{th}} = h(T_{\text{bord}}(y) - T_{\text{ext}})$. On a donc

$$\delta_x T \simeq \frac{hL_x}{2\lambda}(T_{\text{bord}}(y) - T_{\text{ext}}). \tag{22}$$

Enfin, on utilise le fait que si on est dans le cadre de l'approximation 1D, on peut utiliser $T_{\text{bord}}(y) \simeq T(y)$ donné par l'équation (20), soit (avec $\beta \gg 1$ pour simplifier) : $T_{\text{bord}}(y) - T_{\text{ext}} \simeq (T_0 - T_{\text{ext}})\mathrm{e}^{-y/\delta_p}$. On aboutit alors à

$$\delta_x T \simeq \frac{hL_x}{2\lambda}(T_0 - T_{\text{ext}})\mathrm{e}^{-y/\delta_p}. \tag{23}$$

$\delta_x T$ est donc maximal en $y = 0$, avec une valeur $(\delta_x T)_{\max} = \dfrac{T_0 - T_{\text{ext}}}{\alpha^2}$. Il faut donc bien $\alpha^2 \gg 1$ pour utiliser l'approximation 1D qui néglige la variation latérale $\delta_x T$.

2.5.2 Modèle 2D, résolution numérique

Passons maintenant à la résolution numérique 2D. Exprimées en unités du code, les grandeurs pertinentes sont :

- $a = \dfrac{h\delta}{\lambda}$ qui intervient dans les conditions aux bords conducto-convectives du type de l'équation (15).
- $\tilde{\delta}_p = \dfrac{\delta_p}{\delta} = \sqrt{\dfrac{N_x - 1}{2a}}$ la profondeur de peau divisée par la taille d'une cellule. Cette longueur δ_p doit être correctement résolue, donc il faut $\tilde{\delta}_p$ de l'ordre de 50 au moins.
- $\tilde{L}_x = \dfrac{L_x}{\delta} = N_x - 1$ la longueur latérale de l'ailette divisée par la longueur d'une cellule. La structure selon x est capitale car c'est elle qui décrit le flux thermique latéral, il faut donc la résoudre et prendre $\tilde{L}_x$ de l'ordre de 20 au moins.
- $\beta = \dfrac{L_y}{\delta_p} = \dfrac{N_y - 1}{\tilde{\delta}_p} = \sqrt{2a}\dfrac{N_y - 1}{\sqrt{N_x - 1}}$. Peu importe que β soit grand ou non, car l'ailette n'a pas à être infinie (on a l'expression pour L_y quelconque avec le modèle analytique).
- $\alpha = \dfrac{\lambda}{\delta_p h} = \dfrac{1}{a\tilde{\delta}_p} = \sqrt{\dfrac{2}{a(N_x - 1)}}$. On doit avoir $\alpha^2 \gg 1$ si on souhaite comparer le modèle analytique 1D et la simulation numérique 2D. Toutefois si on ne souhaite pas faire cette comparaison, on peut choisir arbitrairement α pour la simulation numérique.

  Indépendamment de α, le caractère 1D est aussi impacté par le ratio L_x/L_y, car le profil de température dans les coins (donc proche de $y = L_y$) est purement 2D : il ne faut pas s'attendre à un excellent accord dans cette zone. On constate ainsi des écarts modèle analytique 1D - simulation 2D qui augmentent lorsque le ratio L_x/L_y devient supérieur à $1/2$.

La figure 10 montre un exemple avec $N_x \times N_y = 32 \times 100$ et $\delta = 1\,\text{cm}$ (soit une barre de 32 cm par 1 m), $T_0 = 100\,°\text{C}$, $T_{\text{ext}} = 10\,°\text{C}$, et les mêmes valeurs de λ et h qu'à la partie 2.4. On a

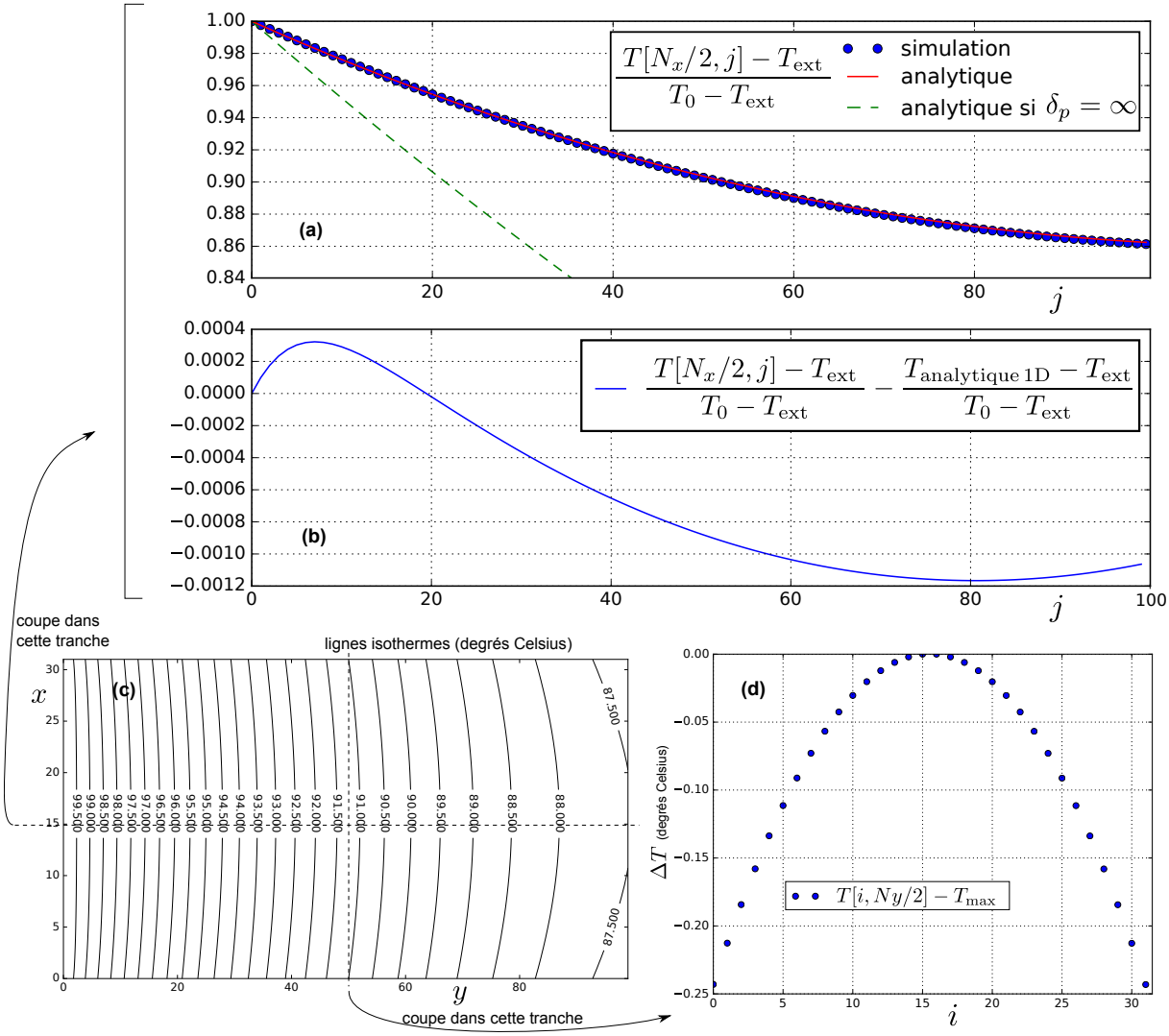


FIGURE 10 – Exemple de résultat pour la simulation d'une ailette de refroidissement.

alors $\alpha = 13$, $\beta = 0.49$, $\tilde{\delta}_p = 203$. Le graphique correspond à $\varepsilon = 10^{-5}$ °C calculé comme l'écart maximal des $|T_k[i,j] - T_{k-1}[i,j]|$.

Quelques remarques :

— L'accord entre solution analytique (équations (20) et (21)) et numérique est très bon.

— Il a fallu environ 5 000 itérations pour avoir un écart maximal $e \leq \varepsilon = 10^{-5}$ °C. On constate que l'écart maximal e décroît exponentiellement avec le nombre d'itérations.

— L'écart entre solution analytique et numérique décroît d'abord avec ε pour ε allant de 10^{-1} °C à 10^{-4} °C, puis reste constant pour $\varepsilon \leq 10^{-5}$ °C en se stabilisant à une valeur relative de 1.2×10^{-3} (écart entre $(T - T_0)/(T_0 - T_{\text{ext}})$ et son homologue analytique, cf figure 10(b)).

La raison de cette non-variation est qu'ici l'écart entre modèle analytique 1D et modèle numérique 2D n'est plus dominé par la convergence de l'algorithme, mais par les approximations qui permettent d'utiliser ou non le modèle 1D : notamment la valeur finie de α, les effets au bout de l'ailette, la résolution finie dans une section.

Un autre essai avec $\delta = 1$ mm (soit une barre de 32 mm par 10 cm, et $\alpha = 41$, $\beta = 0.15$, $\tilde{\delta}_p = 643$) donne des graphiques tout à fait similaires et mène aux mêmes remarques, avec un meilleur accord car l'écart relatif est au maximum de 1.8×10^{-4}. Ceci est à lier au paramètre α qui est plus grand que dans le cas précédent et donc l'hypothèse 1D meilleure.

2.5.3 Conclusion

En conclusion on peut dire que l'accord est très bon. Ceci permet de valider l'algorithme de résolution et l'implémentation des conditions aux bords, le tout pouvant ensuite être utilisé pour des cas où le modèle analytique 1D ne s'applique pas, c'est-à-dire pour α et des ratios L_x/L_y quelconques ou encore pour un coefficient h non uniforme : c'est là tout l'intérêt des simulations numériques.

On peut prolonger l'étude en exploitant davantage les résultats numériques, par exemple en calculant le flux thermique total dissipé par l'ailette afin de donner son efficacité e_{eff}. Cette dernière est définie comme le flux thermique extrait en $y = 0$ en présence de l'ailette, divisé par le flux thermique extrait en l'absence de l'ailette : dans le modèle analytique 1D on aboutit à $e_{\text{eff}} = \alpha(A_1 - B_1)$, mais le modèle numérique permet de l'obtenir dans des cas où l'hypothèse 1D n'est plus valable.

Notons enfin que le modèle retenu ici est très simple, et que le dimensionnement des dissipateurs thermiques fait l'objet de toute une littérature et de modèles poussés.

2.6 Pont thermique

Nous terminons cet article par un exemple qui illustre la souplesse d'utilisation des conditions aux limites et qui montre que le domaine de simulation n'a pas à être rectangulaire. Nous simulons une situation de pont thermique dans un bâtiment à étages : un mur en béton est très bien isolé du côté intérieur de la maison, mais une dalle en béton entre deux étages rompt cette isolation. Il s'agit d'une situation purement 2D qui justifie l'emploi de simulations numériques.

Dans l'exemple montré figure 11 on a pris $\lambda = 2\,\mathrm{W}\cdot\mathrm{K}^{-1}\cdot\mathrm{m}^{-1}$ (valeur typique pour du béton) et à nouveau $h = 15\,\mathrm{W}\cdot\mathrm{m}^{-2}\cdot\mathrm{K}^{-1}$. Le pas est $\delta = 1\,\mathrm{cm}$, et le mur et la dalle ont une épaisseur de 50 cm. Les conditions aux limites sont telles que spécifiées sur la figure.

La taille du domaine est $N_x \times N_y = 300 \times 180$, soit 3 m par 1.80 m. On initialise $B[i, j]$ à 1 sur le tour du mur et de la dalle. Nous avons ajouté une nouvelle valeur possible pour le tableau $B[i, j]$: si $B[i, j] = 2$ alors le point n'est pas à traiter, ce qui est le cas pour toutes les points qui correspondent à l'intérieur de la maison. Le code complet est disponible dans le fichier joint.

On peut ensuite exploiter cette simulation en traçant le profil de température sur le côte extérieur de la maison et en comparant à des relevés thermiques réels, en calculant le flux thermique total perdu, en cherchant à justifier un modèle simplifié en terme de résistances thermiques, en évaluant la température dans les coins de la dalle dans la maison, etc.

Conclusion

Nous avons présenté une méthode de résolution de l'équation de Laplace de complexité (à la fois numérique et de mise en œuvre !) raisonnable pour des étudiants en cycle Licence ou en CPGE, ainsi que l'implémentation de diverses conditions aux bords et de routines d'affichage graphique en Python. Certains des exemples ont été l'occasion de comparaisons avec des solutions analytiques et ont permis de valider le bon fonctionnement de l'ensemble. Le tout peut donc être utilisé dans des situations où les résultats analytiques ne sont pas faciles à formuler – ailette de refroidissement de taille quelconque, pont thermique, condensateur, et bien d'autres exemples dans divers domaines – ce qui est justement tout l'intérêt des simulations numériques.

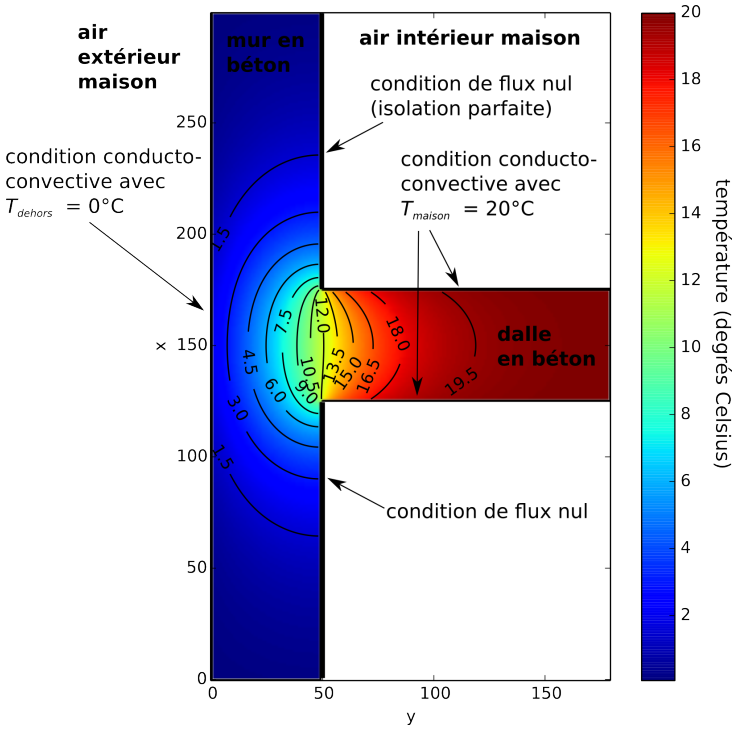


FIGURE 11 – Exemple de simulation d'un pont thermique dans un bâtiment.

Annexe 1 : Compléments

Évaluation du gradient de f

Il peut être utile de calculer le gradient de f, car celui-ci donne le champ électrique, le flux thermique, le flux de particules ou la vitesse selon que $f = V$, T, n ou ϕ. On peut simplement utiliser un schéma centré d'ordre 2 :

$$\frac{\partial f}{\partial x}(x_i, y_j) = \frac{f[i+1, j] - f[i-1, j]}{2\delta} + \mathcal{O}(\delta^2) \tag{24}$$

$$\frac{\partial f}{\partial y}(x_i, y_j) = \frac{f[i, j+1] - f[i, j-1]}{2\delta} + \mathcal{O}(\delta^2) \tag{25}$$

Les évaluations en $i \pm 1$ et $j \pm 1$ ne posent aucun problème car le couple (i, j) n'appartient pas au bord du domaine.

Généralisation à l'équation de Poisson

L'équation de Poisson s'écrit

$$\Delta f + g = 0 \tag{26}$$

avec g une fonction connue. Elle permet de généraliser les cas cités dans l'introduction (potentiel V en présence d'une distribution de charges, température en présence de sources, gravitation dans une distribution de masse, etc.).

Numériquement, il suffit d'ajouter le terme $g(x_i, y_j) \times \delta^2/4$ dans les membres de droite des équations (3), (4), (10), et le terme $g(x_i, y_j) \times \omega\delta^2/4$ dans le membre de droite de l'équation (11). Si on prend l'exemple de cette dernière, elle devient :

$$\begin{aligned} f_k[i, j] &= (1 - \omega) f_{k-1}[i, j] \\ &+ \omega \frac{f_{k-1}[i+1, j] + f_k[i-1, j] + f_{k-1}[i, j+1] + f_k[i, j-1]}{4} + \omega \frac{g[i, j]\delta^2}{4}. \end{aligned} \tag{27}$$

Les résultats théoriques annoncés en 1.4 restent valides, et en particulier l'expression de ω optimal.

Généralisation à une conductivité non uniforme

Les équations de Laplace et de Poisson sont des exemples très particuliers d'équations elliptiques, ces dernières pouvant comporter des coefficients non-constants ou bien être non-linéaires.

Par exemple lorsque la conductivité thermique $\lambda(x, y)$ d'un milieu solide n'est pas uniforme, l'équation de la chaleur en régime stationnaire s'écrit (à deux dimensions) :

$$\frac{\partial}{\partial x}\left(\lambda\frac{\partial f}{\partial x}\right) + \frac{\partial}{\partial y}\left(\lambda\frac{\partial f}{\partial y}\right) = 0. \tag{28}$$

La façon la plus simple de discrétiser cette équation est d'utiliser des différences finies centrées spatialement :

$$\begin{aligned}\frac{\partial}{\partial x}\left(\lambda\frac{\partial f}{\partial x}\right) &\simeq \frac{1}{\delta}\left[\lambda(x_{i+1/2}, y_j)\,\frac{\partial f}{\partial x}(x_{i+1/2}, y_j) - \lambda(x_{i-1/2}, y_j)\,\frac{\partial f}{\partial x}(x_{i-1/2}, y_j)\right] \\ &= \frac{1}{\delta^2}\Big[\lambda[i+1/2, j]\,(f[i+1, j] - f[i, j]) - \lambda[i-1/2, j]\,(f[i, j] - f[i-1, j])\Big],\end{aligned} \tag{29}$$

avec $x_{i+1/2} \equiv (x_i + x_{i+1})/2$ et par exemple $\lambda[i+1/2, j] \equiv (\lambda[i+1, j] + \lambda[i, j])/2$. On fait de même pour le terme en dérivée selon y, et on isole $f[i, j]$ comme pour la méthode de Jacobi. On obtient alors :

$$f[i, j] = \frac{\lambda[i+1/2, j]f[i+1, j] + \lambda[i-1/2, j]f[i-1, j] + \lambda[i, j+1/2]f[i, j+1] + \lambda[i, j-1/2]f[i, j-1]}{\lambda[i+1/2, j] + \lambda[i-1/2, j] + \lambda[i, j+1/2] + \lambda[i, j-1/2]}. \tag{30}$$

On retrouve donc l'expression (3) qui a servi de point de départ à notre exposé, sauf que la moyenne des $f[i \pm 1, j \pm 1]$ est pondérée par les coefficients λ. On peut donc implémenter l'algorithme de la même façon que précédemment, que cela soit pour la version de Jacobi, de Gauss-Seidel ou avec sur-relaxation.

Attention toutefois : il s'agit là plus d'une recette qui semble fonctionner (nous l'avons testée sur le cas du pont thermique, § 2.6, en ajoutant une couche d'isolant de λ élevé contre le mur) que d'un schéma numérique robuste pour lequel des résultats mathématiques de convergence existent. En particulier la valeur optimale du paramètre ω n'est plus connue, et ceci peut faire perdre le gain en complexité apporté par la sur-relaxation. Il faut alors tester différents ω. De plus, nous n'avons pas comparé la solution à une solution analytique. Ce paragraphe est donc ici uniquement pour donner des idées.

Annexe 2 : Exemple d'algorithme

Ci-dessous l'algorithme pour réaliser la simulation de la partie 2.4.2-iii, donc pour un barreau calorifugé sur les côtés, avec T fixé à gauche et une condition de flux conducto-convectif à droite.

Les autres algorithmes, associés à chacune des parties de cet article, sont disponibles dans le fichier Python joint.

```python
def graphe_equipot(B,f,a=0,nb_equipot=25):
    """Affiche les bords du domaine B en noir, et des surfaces équipotentielles de f.
    Prendre a=1 si on souhaite également afficher un colorplot des valeurs de f."""
    Nx,Ny = B.shape
    plt.figure()
    plt.imshow(B,origin='lower',cmap='binary',interpolation='nearest')
    if (a==1):
        plt.imshow(f,origin='lower')  # Pour tracer un colorplot de f
        plt.colorbar()                # idem
    x = np.linspace(0,Nx-1,Nx)        # Array de 0 à Nx-1 inclus en Nx points
    y = np.linspace(0,Ny-1,Ny)
    cont=plt.contour(y,x,f,nb_equipot,colors='k')  # Trace les équipotentielles
    plt.title('equipotentielles')
    plt.clabel(cont,fmt='%1.1f') # Le 1.1 controle le nombre de chiffres après la
    plt.show()                                          # virgule sur l'affichage

def w_opt(Nx,Ny):  # Calcul du paramètre omega optimal
    return 2./(1.+np.pi/(Nx*Ny)*((Nx**2+Ny**2)/2.)**0.5)

def initialisation_cadre(B,T,T0):
    """Initialise le domaine et ses bords. B matrice des bords,
    T tableau (fonction f), T0 température imposée à gauche)."""
    B[:,0] = 1      # La matrice B
    B[:,Ny-1] = 1     # est initialisée
    B[0,:] = 1        # à 1 sur
    B[Nx-1,:] = 1     # tout le cadre
    T[:,0] = T0     # Bord gauche, cette valeur ne sera plus modifée (Dirichlet)
    T[:,Ny-1] = T0 # Bord droit
    T[0,:] = T0     # Bord haut
    T[Nx-1,:] = T0 # Bord bas

def une_iteration_GS(B,f):
    """Réalise une itération via la méthode de Gauss-Seidel.
    B matrice des bords, f matrice de la fonction.
    Retourne l'écart (e dans l'article)."""
    ecart_max = 0.
    # actualisation sur tout le domaine, i allant de haut en bas, j de gauche à droite :
    for i in range(Nx):
        for j in range(Ny):
            temp = f[i,j]
            if B[i,j]==0:  # on n'est alors pas sur un bord
                f[i,j] = (1.-w)*temp + w*(f[i+1,j]+f[i,j+1]+f[i-1,j]+f[i,j-1])/4.
            if j==Ny-1 and 0<i<Nx-1:                # droite du cadre, coins exclus.
                f[i,j] = (f[i,j-1]+a*Text)/(1.+a)  # -> flux fixé par conductoconvectif
            if i==0 and 0<j<Ny-1:                   # haut du cadre, coins exclus.
                f[0,j] = f[1,j]                     # -> flux nul
            if i==Nx-1 and 0<j<Ny-1:                # bas du cadre, coins exclus.
                f[Nx-1,j] = f[Nx-2,j]               # -> flux nul
            ecart_max = max(ecart_max, abs(temp-f[i,j]))
    f[Nx-1,Ny-1] = 0.5*(f[Nx-2,Ny-1]+f[Nx-1,Ny-2]) # traiter les coins (inutile mais
    f[0,Ny-1]    = 0.5*(f[1,Ny-1]+f[0,Ny-2])       # plus joli sur le rendu graphique)
    return ecart_max

def iterations_GS(B,f,eps):
    """Enchaine les itérations jusqu'à ce que le critère de convergence eps soit atteint.
    B matrice des bords, f matrice de la fonction, eps = epsilon dans l'article."""
    ecart = eps+1.
    nb_iterations = 0
    while ecart > eps:
        ecart = une_iteration_GS(B,f)
        nb_iterations += 1
    return nb_iterations

Nx, Ny = 10, 100  # taille du domaine
w = w_opt(Nx,Ny)  # calcul du paramètre de sur-relaxation omega optimal
Text = 10.        # degrés, température exterieure
T0 = 100.         # degrés, température de la face à gauche
Tinitiale = T0    # degrés, température initiale du barreau
delta = 1.e-2     # pas de la grille en m
lambd = 400.      # W.K^-1.m^-1, conductivité thermique
h = 15.           # W.m^-2.K^-1, coefficient conducto-convectif de Newton
a = delta*h/lambd # paramètre intervenant dans les CL conducto-convectives
B = np.zeros((Nx,Ny),bool)          # matrice des bords (0 ou 1)
T = np.zeros((Nx,Ny)) + Tinitiale   # matrice contenant la température
initialisation_cadre(B,T,T0)        # on initialise le domaine
iterations_GS(B,T,1.e-7)            # on effectue les itérations, epsilon=1.e-7 ici
graphe_equipot(B,T)               # tracé des équipotentielles
```

Remerciements

Merci à Étienne Thibierge pour sa relecture de l'article et ses suggestions, et à Alain Bernard pour l'idée du pont thermique. Je remercie également le relecteur pour ses corrections qui ont permis d'améliorer l'article.

Références

[1] W. H. Press, S. A. Teukolsky, Vetterling W. T., and Flannery B. P. *Numerical Recipes in Fortran (2nd ed) : the art of scientific computing.* Cambridge University Press, 1992.

[2] M. Chapelet. Résolution de l'équation de Laplace en électrostatique à l'aide d'un micro-ordinateur. *Bull. Un. Phys.*, 672(1) : 767–780, March 1985.

[3] P. B. Hansen. Numerical Solution of Laplace's Equation. *Electrical Engineering and Computer Science Technical Reports*, Paper 168, 1992.

[4] S. Cahuzy. De l'orage dans l'air. *Bull. Un. Phys.*, 913 : 369–384, 2009.

[5] J. D. Jackson. *Classical Electrodynamics, 3rd Edition.* July 1998.