TD n° 5 : Équations de Laplace et de Poisson

1 Généralités

En Physique, on rencontre très souvent des équations différentielles partielles telles que les équations d'onde, de Navier-Stockes, de diffusion ou bien de la chaleur. Le but de ce TD est de résoudre numériquement l'équation de Poisson [1] à deux dimensions :

$$\Delta u \;=\; -\rho(x,y) \qquad \Leftrightarrow \qquad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \;=\; -\rho(x,y). \tag{1}$$

Quand le terme de sources ρ est nul, l'équation se réduit à une équation de Laplace :

$$\Delta u \;=\; 0 \qquad \Leftrightarrow \qquad \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \;=\; 0. \tag{2}$$

La résolution numérique des équations (1) et (2) nécessite la connaissance des conditions vérifiées par la fonction u ou certaines de ses dérivées aux bords du domaine considéré. On parle de *conditions de Dirichlet* quand ces conditions ne portent que sur les valeurs de la fonction u et de *conditions de Neumann* quand ces conditions ne portent que sur les valeurs de la dérivée normale de u. On peut aussi avoir des conditions mixtes.

Pour résoudre numériquement l'équation de Poisson, on représente la fonction u sur une grille cartésienne carrée de taille $a \times a$ discrétisée en $(J_{MAX}+1)\times(J_{MAX}+1)$ noeuds, et de pas spatial δ (voir Fig. 1). Les abscisses x_j et les ordonnées y_l sont discrétisées :

$$x_j \;=\; x_0 + j\delta \qquad \text{avec} \quad j = 0,1,2,...,J_{MAX}, \tag{3}$$

$$y_l \;=\; y_0 + l\delta \qquad \text{avec} \quad l = 0,1,2,...,J_{MAX}. \tag{4}$$

L'équation (1) est évaluée en utilisant des différences finies. En posant $u_{j,l} = u(x_j, y_l)$, il vient

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \approx \frac{u_{j+1,l} - 2u_{j,l} + u_{j-1,l}}{\delta^2} \;+\; \frac{u_{j,l+1} - 2u_{j,l} + u_{j,l-1}}{\delta^2} = -\rho_{j,l}, \tag{5}$$

soit

$$u_{j+1,l} \;+\; u_{j-1,l} \;+\; u_{j,l+1} \;+\; u_{j,l-1} \;-\; 4u_{j,l} \;=\; -\rho_{j,l}\,\delta^2. \tag{6}$$

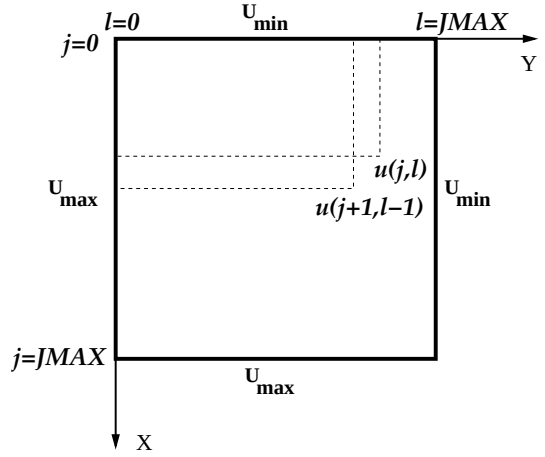
On obtient ainsi un système d'équations dont les conditions aux bords peuvent être imposées en fixant la valeur de certains termes $u_{j,l}$: le problème initial est réduit à la résolution d'un système linéaire (voir question 14). Ce n'est pas la méthode qui sera utilisée dans la quasi-totalité de ce TD. Nous allons utiliser la méthode dite de *relaxation*. C'est un processus itératif dans lequel la solution recherchée $u(x,y)$ peut être vue comme la solution stationnaire d'un problème de diffusion :

$$u(x,y) \;=\; \lim_{t\to\infty} \tilde{u}(x,y,t) \qquad \text{avec} \qquad \frac{\partial \tilde{u}}{\partial t} = \Delta\tilde{u} + \rho(x,y). \tag{7}$$

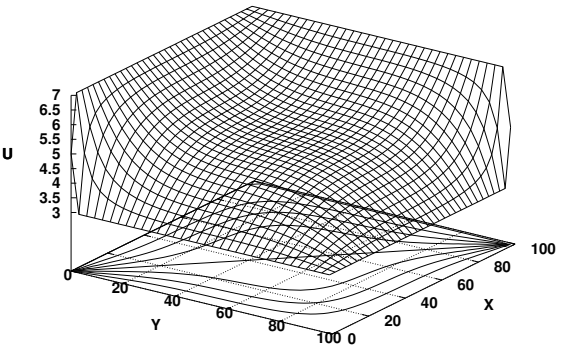
1. Un autre exemple type d'équation aux dérivées partielles est l'équation des ondes :

$$\Delta u \;=\; \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}.$$

où c est la célérité de l'onde.



(a) Schéma de principe de la grille de discrétisation de l'espace avec les conditions au bords spécifiques décrites dans l'énoncé.



(b) Exemple de solution de l'équation de Laplace par la méthode de Jacobi avec des conditions de Dirichlet non nulles sur les quatres bords. Des courbes de niveaux équidistants sont représentées sur la base.

FIGURE 1 – Résolution de l'équation de Laplace sur une grille cartésienne 2D.

Dans un schéma de différences finies on peut noter dt le pas de temps et n l'indice de comptage qui lui est associé ($t = n\,dt$). On écrit alors $\tilde{u}(x, y, n\,dt) = u^n_{j,l}$, et l'équation (7) s'écrit :

$$\frac{u^{n+1}_{j,l} - u^n_{j,l}}{dt} \;=\; \frac{1}{\delta^2}(u^n_{j+1,l} + u^n_{j-1,l} + u^n_{j,l+1} + u^n_{j,l-1} - 4u^n_{j,l}) + \rho_{j,l}. \tag{8}$$

On peut montrer que le pas de temps $dt = \delta^2/4$ donne un schéma de différences finies stable. En utilisant cette valeur, l'équation (8) devient

$$\boxed{u^{n+1}_{j,l} \;=\; \frac{1}{4}(u^n_{j+1,l} + u^n_{j-1,l} + u^n_{j,l+1} + u^n_{j,l-1}) + \frac{\delta^2}{4}\rho_{j,l}.} \tag{9}$$

Cette équation se résout de manière itérative par rapport au temps. La valeur de $u^{n+1}_{j,l}$ au nœud (j, l) de la grille à l'instant $t = (n+1)\,dt$ (*i.e.* itération $n+1$) est obtenue en sommant la contribution du terme source ρ à la valeur moyenne spatiale de u sur les quatre nœuds les plus proches. Dans la *méthode de Jacobi*, la moyenne spatiale est faite avec les valeurs à l'étape n. Elle converge en fait lentement. On peut l'améliorer en calculant la moyenne de u aux quatre nœuds voisins, non pas à l'étape n, mais en actualisant au fur et à mesure les éléments de la grille. Cette manière de procéder est aussi plus facile à programmer puisqu'une seule valeur de u par site doit être stockée à chaque itération. Elle se nomme *méthode de Gauss-Seidel*.

2 Travail demandé

On va résoudre l'équation de Laplace en utilisant l'équation (9) sur une grille carrée $(J_{MAX} + 1) \times (J_{MAX} + 1)$ pour un terme source ρ nul. Dans ce cas la valeur de δ ne joue pas de rôle dans la formule (9).

1. Déclarer un tableau bi-dimensionnel `u` de la bonne taille.
2. Imposer les conditions aux bords suivantes (voir Fig. 1(a)), en définissant les constantes appropriées en début de programme :

(a) bord inférieur ($X = \delta J_{MAX}$ *i.e.* $j = J_{MAX}$) : $u = 0$

(b) bord supérieur ($X = 0$ *i.e.* $j = 0$) : $u = 0$

(c) bord gauche ($Y = 0$ *i.e.* $l = 0$) : u=0

(d) bord droit ($Y = \delta J_{MAX}$ *i.e.* $l = J_{MAX}$) : u=`U_MAX`

(e) Pour les coins, on prendra la valeur moyenne entre les deux valeurs adjacentes.

Prendre `U_MAX` $= 3.0$.
Pour les conditions initiales, procéder ainsi :
$u^0_{jl} =$ `U_INIT` avec `U_INIT` = `U_MAX`$/2$, pour $(j, l) \in]0, J_{MAX}[\times]0, J_{MAX}[$.

3. Programmer tout d'abord la *méthode de Jacobi*. Pour cela, déclarer deux tableaux bi-dimensionnels d'éléments de type `float`, `u_ancien` et `u` qui serviront à stocker les valeurs des sites de la grille à l'étape n et $n + 1$. Faire afficher le nombre n de fois où la boucle de mise à jour est parcourue. On fixera le nombre `NMAX` d'itérations (pour vos essais prendre `NMAX` de l'ordre de 10) et utiliser une grille telle que `JMAX` $= 25$.
4. Visualiser la grille 2D avec l'instruction `pcolormesh` du module `pylab`. Sauver l'image au format png dans le fichier `grille_laplace_NMAX100.png`.
5. En gardant les mêmes conditions aux bords et initiales, utiliser maintenant un critère de convergence pour arrêter le processus itératif. On peut prendre comme critère :

$$\sup_{0<j,l<J_{MAX}} (|u^{n+1}_{jl} - u^n_{jl}|) < \epsilon.$$

Fixer la valeur de ϵ à l'aide de la constante `EPS`, on prendra pour la suite `EPS`=0.01 et $J_{MAX} = 40$.

6. Visualiser la grille 2D avec `pcolormesh` du module `pylab`. Sauver l'image au format png dans le fichier `grille_laplace_epsilon.png`.
7. Dans un nouveau fichier, programmer la *méthode de Gauss-Seidel* et comparer le nombre d'itérations nécessaires à la convergence pour les deux méthodes pour une même grille et les mêmes conditions aux bords et initiales. Indiquer vos résultats et conclusions en commentaire dans votre programme. Tester aussi les deux méthodes pour $JMAX = 100$.
8. En fait, le problème précédant admet un solution analytique qui n'est pas conceptuellement difficile à obtenir (séparation de variables, transformée de Fourier en sinus et cosinus, ...) mais est un peu calculatoire. On obtient la série suivante pour la solution analytique notée $u_{analy}(x, y)$ telle que $u_{analy}(x, y = a) =$ `U_MAX` et u_{analy} soit nulle sur les autres bords de la grille :

$$\boxed{u_{analy}(x, y) \;=\; \frac{4\,U_{MAX}}{\pi} \sum_{k=1}^{\infty} \frac{\sin(n\pi x/a)\sinh(n\pi y/a)}{n\sinh(n\pi)} \text{ avec } n = 2k-1} \qquad (10)$$

Dans un nouveau fichier, rajouter le calcul de cette solution. On se limitera au 20 premiers termes de la série (cf Eq. (10)) puis au 40. Déclarer le tableau `u_analy` de taille $(J_{MAX} + 1) \times (J_{MAX} + 1)$ qui contiendra l'évaluation analytique sur la grille discrétisée. Remplir en appliquant la formule (10), et représenter graphiquement cette solution avec l'instruction `pcolormesh`.

9. A partir de cette question, et pour assurer une meilleure correspondance avec la description analytique de la solution, on fixera u à U_{MAX} sur tout le bord droit y compris les deux coins haut et bas. Représenter graphiquement la différence entre les solutions numérique et analytique sur l'ensemble de la grille. Sauver la grille obtenue sous `difference_relaxa_analy.png`. Commentez.
10. Pour $J_{MAX} = 100$, déterminer puis tracer l'erreur maximale commise sur l'ensemble de la grille en fonction de ϵ. Que concluez-vous ?
11. Selon vous, quelle serait la solution analytique, si les conditions sur les quatre bords étaient maintenant : $u_{analy}(x, y = a) = \mathtt{U_{MAX}} + \mathtt{U_{DECAL}}$, $u_{analy}(x, y = 0) = \mathtt{U_{DECAL}}$, $u_{analy}(x = a, y = 0) = \mathtt{U_{DECAL}}$, $u_{analy}(x = 0, y) = \mathtt{U_{DECAL}}$? Justifier votre réponse. Inscrire votre réponse en commentaire dans votre programme.
12. Calculer numériquement cette nouvelle solution pour $\mathtt{U_{DECAL}} = 10$ en utilisant la méthode de Jacobi ou celle de Gauss-Siedel, sauver la grille obtenue sous `grille_laplace_num2.png`. Commentez dans votre fichier source.
13. On peut améliorer la précision du calcul du laplacien dans l'équation (5), en passant d'une approximation en δ^2 à une en δ^4.

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \approx & \quad \frac{1}{\delta^2}\left(-20u^n_{j,l} + 4\left(u^n_{j+1,l} + u^n_{j-1,l} + u^n_{j,l+1} + u^n_{j,l-1}\right)\right. \\ & + \left. u^n_{j+1,l+1} + u^n_{j-1,l-1} + u^n_{j+1,l-1} + u^n_{j-1,l+1}\right) = -\rho_{j,l}\end{aligned} \tag{11}$$

L'équation (9) devient alors :

$$\boxed{u^{n+1}_{j,l} = \frac{1}{20}\left(4\left(u^n_{j+1,l} + u^n_{j-1,l} + u^n_{j,l+1} + u^n_{j,l-1}\right) + u^n_{j+1,l+1} + u^n_{j-1,l-1} + u^n_{j+1,l-1} + u^n_{j-1,l+1}\right) + \frac{\delta^2}{4}\rho_{j,l}.} \tag{12}$$

Dans un nouveau fichier, rajouter le calcul de cette solution dans la nouvelle fonction pour le cas $\rho = 0$. Pour les mêmes conditions aux bords et initiales, et la même taille de grille $\mathtt{JMAX} = 25$, indiquer en commentaire dans votre programme, le nombre d'itérations requises par les deux approximations du laplacien (en δ^2 pour l'équation (5) et en δ^4 pour l'équation (11)) dans le cadre de la méthode de Jacobi pour atteindre un critère de convergence $\epsilon < 0.001$. Commenter. Quel est l'intérêt de la nouvelle approximation ?
14. *Méthode directe :* Uniquement si vous avez le temps, revenez à l'équation (6). Comme indiqué, on a un système linéaire à résoudre. Ecrivez le système linéaire à résoudre (demander si besoin est le document supplémentaire). Comment le caractériser ?
En utilisant les fonctions des modules `Python` (`scipy.lingalg` ou `scipy.sparse.linalg`, tester les deux et comparer), résoudre directement le système et comparer la solution obtenue avec la soluton analytique du problème (voir la question 8). Représenter graphiquement la différence entre la nouvelle solution numérique et la solution analytique.
En ayant rajouté l'évaluation du temps de calcul pour chaque méthode concernée, comparer la méthode directe avec la dernière méthode de relaxation codée à la question précédente. Qu'en conluez-vous ?