PPSC2020F - Projet I

J.PERDIGON

8 février 2021

1 Introduction

On considère le problème de Poisson à 2 dimensions avec des conditions aux bords de Dirichlet

$$\begin{aligned} -\Delta_{\mathbf{x}} u(\mathbf{x}) = f(\mathbf{x}) \ , \ \mathbf{x} \in \Omega = (0, L_x) \times (0, L_y) \\ u(\mathbf{x}) = \alpha(\mathbf{x}) \ , \ \mathbf{x} \in \partial\Omega \end{aligned} \tag{1}$$

Dans un premier temps, on peut réécrire le problème en effectuant les changements de variables $\tilde{x} = \frac{x}{L_x}$, $\tilde{y} = \frac{y}{L_y}$. L'opérateur de Laplace se réécrit ainsi

$$-\Delta_{\mathbf{x}} u(\tilde{\mathbf{x}}) = \frac{1}{L_x^2}\partial_{\tilde{x}}^2 u + \frac{1}{L_y^2}\partial_{\tilde{y}}^2 u \tag{2}$$

On choisit d'approximer le problème à l'aide de différences finies. L'espace est discrétisé en une grille uniforme de $(n+1)$ points dans chaque direction avec le pas de grille $h = 1/n$. L'opérateur Laplacien Δ est quant à lui approximé par :

$$\Delta_{\mathbf{x}} u(\tilde{\mathbf{x}}) \approx \frac{1}{L_x^2}\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{1}{L_y^2}\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} \tag{3}$$

Où $u_{i,j}$ représente une approximation discrete de la solution, sur la grille d'espace définie précédemment. $u_{i,j}$ peut être représenté sous la forme d'une matrice U, de taille $m \times m$ ($m = (n-1)$) et dont les indices i et j sont respectivement les indices de lignes et de colonnes. Le problème se réécrit sous forme matricielle

$$\frac{1}{L_x^2}TU + \frac{1}{L_y^2}UT = h^2 F \tag{4}$$

Où U et T sont définies comme :

$$U = \begin{pmatrix} u_{1,1} & \cdots & \cdots & \cdots & u_{1,n-1} \\ \vdots & \ddots & & & \vdots \\ \vdots & & u_{i,j} & & \vdots \\ \vdots & & & \ddots & \vdots \\ u_{n-1,1} & \cdots & \cdots & \cdots & u_{n-1,n-1} \end{pmatrix} \ , \ T = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \tag{5}$$

Les matrices T et F sont respectivement la matrice de Laplace unidimensionnelle et la matrice associée au membre de droite. A noter que écrivant le problème sous cette forme matricielle, l'implémentation des conditions aux bords a été absorbée dans la définition de la matrice F

$$F = \begin{pmatrix} f_{1,1} + \frac{\alpha_{0,1}}{L_x^2} + \frac{\alpha_{1,0}}{L_y^2} & f_{1,2} + \frac{\alpha_{0,2}}{L_x^2} & \cdots & f_{1,n-2} + \frac{\alpha_{0,n-2}}{L_x^2} & f_{1,n-1} + \frac{\alpha_{0,n-1}}{L_x^2} + \frac{\alpha_{1,n}}{L_y^2} \\ f_{2,1} + \frac{\alpha_{2,0}}{L_y^2} & & \vdots & & f_{2,n-1} + \frac{\alpha_{2,n}}{L_y^2} \\ \vdots & \cdots & f_{i,j} & \cdots & \vdots \\ f_{n-2,1} + \frac{\alpha_{n-2,0}}{L_y^2} & & \vdots & & f_{n-2,n-1} + \frac{\alpha_{n-2,n}}{L_y^2} \\ f_{n-1,1} + \frac{\alpha_{n,1}}{L_x^2} + \frac{\alpha_{n-1,0}}{L_y^2} & f_{n-1,2} + \frac{\alpha_{n,2}}{L_x^2} & \cdots & f_{n-1,n-2} + \frac{\alpha_{n,n-2}}{L_x^2} & f_{n-1,n-1} + \frac{\alpha_{n,n-1}}{L_x^2} + \frac{\alpha_{n-1,n}}{L_y^2} \end{pmatrix} \tag{6}$$

Pour résoudre le système (4), on peut utiliser une méthode itérative. Cependant, il existe une méthode directe de résolution, en remarquant que l'on connaît les valeurs propres ainsi que les vecteurs propres de l'opérateur de Laplace associé à la matrice T

$$T\ s_i = \lambda_i\ s_i \ , \ s_i = [\sin(i\pi h), \sin(2i\pi h), ..., \sin(mi\pi h)]^T \ , \ \lambda_i = 4\sin(\frac{i\pi h}{2})^2 \ , \ i = 1, ..., m \tag{7}$$

Les vecteurs propres $\{S_i\}$ forment les colonnes de la matrice S. L'équation (7) se réécrit ainsi

$$TS = DS \ , \ S = [s_1^T, s_2^T, ..., s_m^T] \ , \ D = diag(\lambda_1, \lambda_2, ..., \lambda_m) \tag{8}$$

La matrice S est symétrique ($S^T = S$) et orthogonale à un facteur près ($S^T S = S^2 = 1/2h\ I$). Ainsi, en décomposant la solution U dans la base formée par les vecteurs propres de T ($U = S\tilde{U}S$), l'équation (4) se réécrit

$$\frac{1}{L_x^2} TS\tilde{U}S + \frac{1}{L_y^2} S\tilde{U}ST = h^2 F \tag{9}$$

En multipliant à gauche et à droite de chaque terme par la matrice S

$$\frac{1}{L_x^2} STS\tilde{U}S^2 + \frac{1}{L_y^2} S^2\tilde{U}STS = h^2 \tilde{F} \tag{10}$$

Où $\tilde{F} = SFS$. En utilisant $TS = DS = SD$, on obtient

$$\frac{1}{L_x^2} S^2 D\tilde{U}S^2 + \frac{1}{L_y^2} S^2\tilde{U}S^2 D = h^2 \tilde{F} \tag{11}$$

Finalement, on obtient l'équation pour $\tilde{U}$ en utilisant $S^2 = 1/2h\ I$,

$$\frac{1}{L_x^2} D\tilde{U} + \frac{1}{L_y^2} \tilde{U}D = 4h^4 \tilde{F} \ \leftrightarrow \ \tilde{U}_{i,j} = 4h^4 \frac{\tilde{F}_{i,j}}{\frac{\lambda_i}{L_x^2} + \frac{\lambda_j}{L_y^2}} = 4h^4 L_x^2 L_y^2 \frac{\tilde{F}_{i,j}}{L_y^2 \lambda_i + L_x^2 \lambda_j} \tag{12}$$

Ainsi, en se donnant une matrice F de taille $m \times m$, il faut effectuer 4 produits matriciels ($4m^3$ opérations) et un produit de matrices terme à terme (produit d'Hadamard en m^2 opérations) pour obtenir la solution U. La solution du problème de Poisson peut être calculée $\mathcal{O}(n^3)$. Ce nombre peut-être réduit, en remarquant que le produit matriciel de la matrice S avec une matrice A correspond en réalité à la DST des colonnes de cette même matrice A. La DST d'un vecteur $\mathbf{v}$ de taille m peut être déduite de la FFT du vecteur $\mathbf{v_L}$ de taille $2m + 2$ selon

$$DST(\mathbf{v})_k = -\frac{1}{2}\ Imag\left(FFT(\mathbf{v_L})_{k+1}\right) \ , \ \mathbf{v_L} = [0, v_0, v_1, ..., v_{m-1}, 0, -v_{m-1}, -v_{m-2}, ..., -v_0]^T \ , \ k = 0, 1, .., m-1 \tag{13}$$

De cette manière, la DST se calcule en $\mathcal{O}(m \ln(m))$ opérations. La solution du problème de Poisson est ainsi calculée en $\mathcal{O}(n^2 \ln(n))$.

2 Implementation numérique

L'implémentation se fait au sein d'une classe "TwoDPoisson" qui prend en arguments n (puissance de 2), L_x et L_y. Le constructeur se charge ensuite de calculer la solution U (la variable privée $u\_$) en cinq étapes :

- $F_1 = S\ F$
- $\tilde{F} = F_1\ S$
- $\tilde{U}_{i,j} = 4h^4 L_x^2 L_y^2 \frac{\tilde{F}_{i,j}}{L_y^2 \lambda_i + L_x^2 \lambda_j}$
- $U_1 = S\ \tilde{U}$
- $U = U_1\ S$

Les produits matriciels $S\ F$ et $S\ \tilde{U}$ sont calculés à l'aide de la fonction de classe privée $dst\_col()$ qui effectue respectivement la DST des colonnes de F et $\tilde{U}$. Les produits matriciels $F_1\ S$ et $U_1\ S$ sont quant à eux calculés à l'aide de la fonction $dst\_row()$ qui effectue la DST des lignes de F_1 et U_1.

2.1 Parallélisation avec MPI

On veut à présent paralléliser la classe "TwoDPoisson". La tâche que l'on souhaite paralléliser est le calcul de de la DST sur les colonnes/lignes de la matrice considérée. Soit P le nombre de processeurs alloué au calcul ; le nombre de colonnes/lignes assigné à chaque processeur sur lesquelles celui-ci calculera la DST est $M = m/P$. Le reste de colonnes/lignes $M_r = m\%P$ est pris en charge par le dernier processeur $P-1$. Afin d'optimiser le nombre de communications entre les processeurs (au détriment de la mémoire), on opte pour la stratégie suivante : chaque processeur $p = 0, ..., P-1$ possède une version de la matrice dont on veut calculer la DST des colonnes/lignes dont il ne calcule cependant que la DST des colonnes/lignes associées aux indices Mp à $M(p+1)$ (le dernier processeur va de $M(P-1)$ à $MP + M_r$). Se faisant, chaque processeur calcule donc la DST d'une sous-partie de

la matrice. Finalement la matrice sur chaque processeur est mise à jour dans son ensemble, grâce aux fonctions "communication_col()" et "communication_row()" où chaque processeur va communiquer aux autres processeurs la sous-partie qu'il aura calculé (via la fonction $MPI\_Allgatherv()$). Une barrière $MPI\_Barrier()$ est ensuite ajoutée en sécurité afin d'attendre que la matrice soit mise à jour sur chaque processeur, avant de passer à d'autres calculs.

3 Cas test

Dans un premier temps, on se propose de tester le code dans le cas simple

$$\begin{aligned} -\Delta_{\mathbf{x}} u(\mathbf{x}) = 1 \ , \ \mathbf{x} \in \Omega = (0,\pi) \times (0,\pi) \\ u(\mathbf{x}) = 0 \ , \ \mathbf{x} \in \partial\Omega \end{aligned} \tag{14}$$

La solution $u(x,y)$ s'écrit sous la forme

$$u(x,y) = \sum_{k=1}^{\infty} \sum_{l=1}^{\infty} U_{kl} \sin(kx) \sin(ly) \ , \ U_{kl} = 4 \frac{(1-(-1)^k)(1-(-1)^l)}{(k^2+l^2)kl\pi^2} \tag{15}$$

Afin de tester la validité du programme, on se proposer de calculer une estimation de l'erreur, noté ϵ_2 et défini par

$$\epsilon_2 = \frac{1}{m} \left[\sum_{i=1}^{m} \sum_{j=1}^{m} \left(u(x_i, y_j) - u_{i,j} \right)^2 \right]^{\frac{1}{2}} \tag{16}$$

ϵ_2 représente la norme L2 de la différence entre $u(x_i, y_j)$ et $u_{i,j}$. Puisque l'on a approximé l'opérateur Laplacien par un opérateur de différences finies d'ordre 2, on s'attend à ce que ϵ_2 décroisse en n^2 avec avec le nombre de points n. Concernant le temps de calcul T_1, comme nous l'avons vu, on s'attend à ce qu'il croisse en $n^2 \ln(n)$ avec n. En Fig.1, on représente ϵ_2 et T_1 en fonction de n. Puisque $u(x,y)$ s'exprime comme une somme infinies de termes, on coupe en pratique la série de manière à ce que l'erreur engendrée soit toujours dominée par l'erreur de troncature ϵ_2.

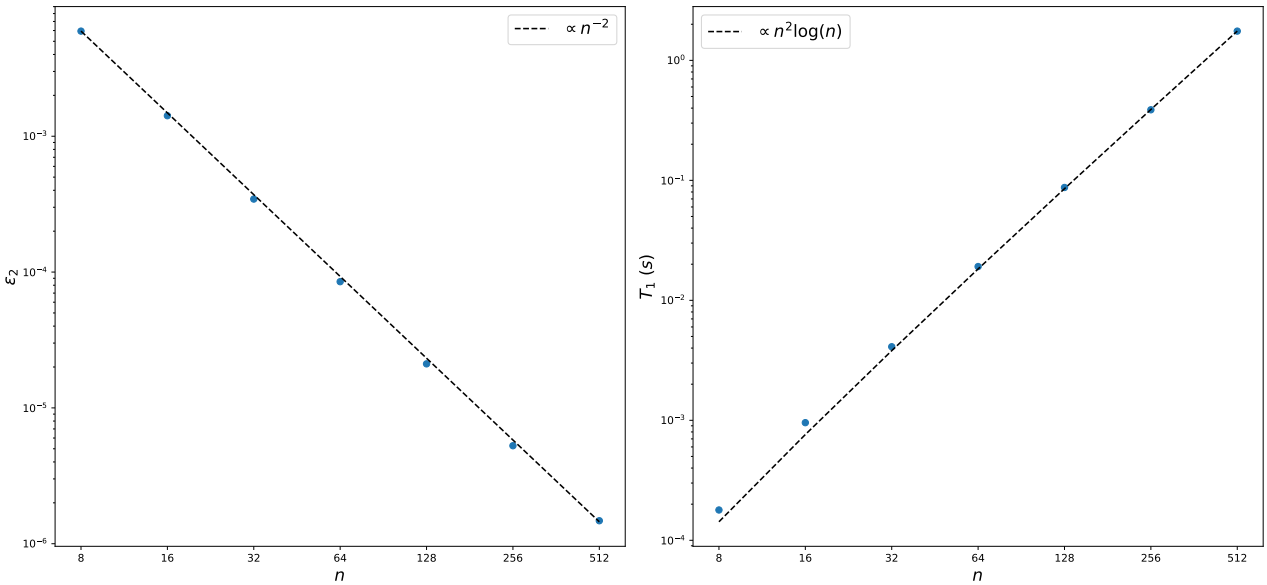

FIGURE 1 – ϵ_2 et T_1 en fonction du nombre de points n, avec $P = 1$. Les courbes en pointillés représentent le comportement analytique attendu.

On se penche à présent sur les performances de la parallélisation. Puisque notre problème est calculé sur l'ensemble des processeurs (cela découle de notre stratégie de parallélisation voir Sect. 2.1), on définit le temps de calcul du problème parallélisé T_P comme maximum des temps sur chaque processeurs ($T_P = max_P(T_p)$). En

Fig. 3 on représente le facteur d'accélération T_1 / T_P en fonction de P, pour différents n. La courbe en pointillées représente le comportement idéal où il n'existe pas de surcoûts liés aux communications entre les processeurs (T_1 / $T_P = n^2 \log(n)$ / $(n^2 \log(n)/P) = P$). Sur le panneau de gauche, on représente les données expérimentales. On remarque que plus le nombre de processeurs P est grand plus on s'éloigne de la courbe idéale, ce qui peut s'expliquer par le fait qu'en augmentant le nombre de processeurs, on augmente le nombre de communications et donc le surcoût. De plus, on constate également que plus n est grand, plus on se rapproche de la courbe idéale. Là encore, cela s'explique par le fait qu'en augmentant n, on diminue en proportion les communications entre processeurs par rapport à la charge de travail. Ces conclusions sont résumées dans le panneau de droite, où l'on représente le facteur d'amplification théorique, en assumant que les surcoûts de communication sont de la forme $\propto (P-1)n^2$. Dans ce cas, un processeur quelconque communique aux autres processeurs une matrice de taille $n \times n$ et le facteur s'exprime comme

$$\frac{T_1}{T_P} = \frac{n^2 \log(n)}{\frac{1}{P} n^2 \log(n) + A(P-1)n^2} \tag{17}$$

Où A est une constante de proportionnalité quelconque.

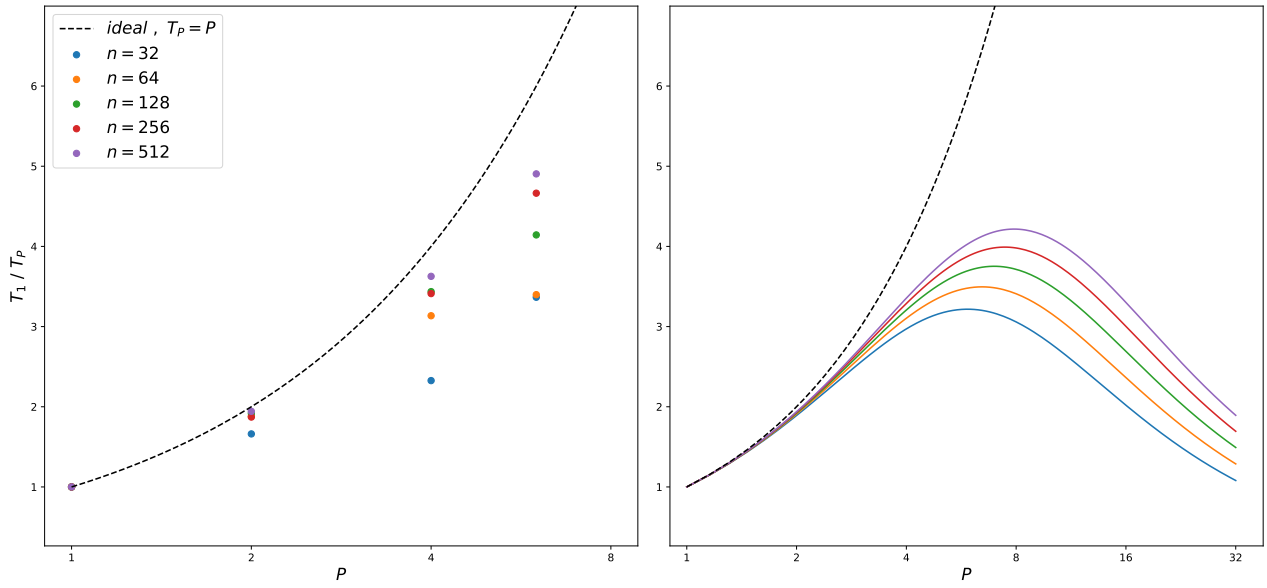

FIGURE 2 – Le facteur d'accélération T_1 / T_P en fonction de P, pour différents n. (Gauche) Données expérimentales. (Droite) Les courbes théoriques en assumant un surcoût de communication $\propto (P-1)n^2$. Les courbes en pointillées représentent le comportement idéal où il n'existe pas de surcoûts liés aux communications entre les processeurs.

4 Quelques exemples

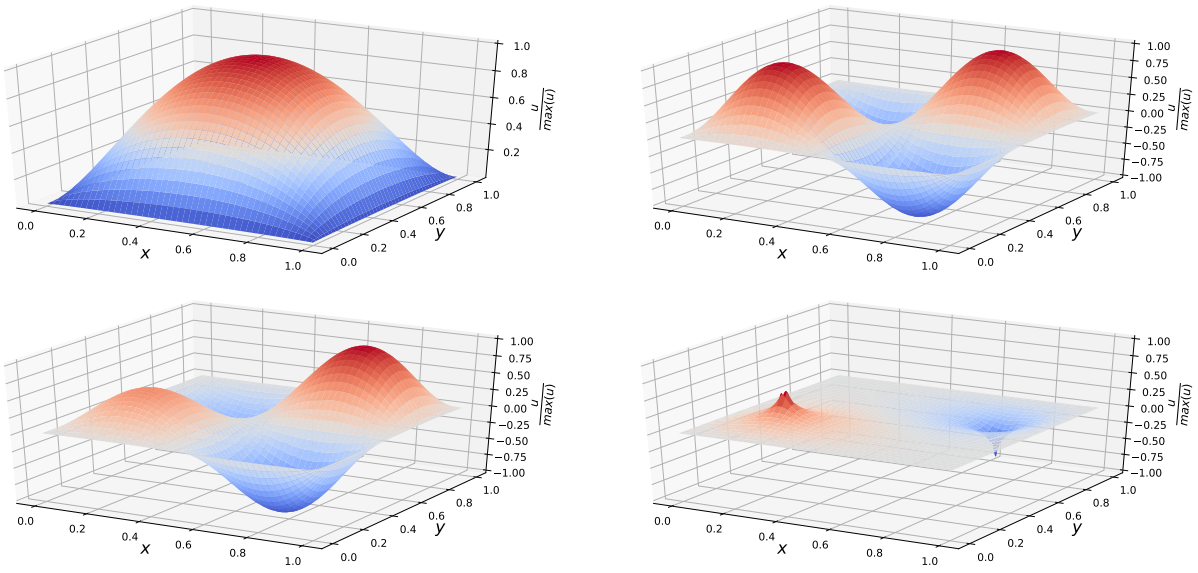

FIGURE 3 – Profil normalisé de l'approximation de u, résultant du code, en fonction des coordonnées d'espace $x, y \in \Omega = (0,1) \times (0,1)$, pour $n = 512$. En haut à gauche : $f(x,y) = 1$, en haut à droite $f(x,y) = \sin(2\pi x)\sin(2\pi y)$, en bas à gauche $f(x,y) = \exp(x)\sin(2\pi x)\sin(2\pi y)$, en bas à droite $f(x,y) = \delta(x-0.25, y-0.25) - \delta(x-0.75, y-0.75)$.