Compte-Rendu CELIK-DRUESNES : L'équation de Poisson

CELIK Baran - DRUESNES Mathias

1 Énoncé de l'équation de Poisson

Soit $\boldsymbol{u}$ solution de l'équation de Poisson; en nommant $\boldsymbol{s}$ le terme source, $\boldsymbol{u}$ est solution de :
$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = s(x, y)$

2 Introduction au problème

2.1 Électrostatique

L'équation de Poisson trouve son application dans le domaine de l'électrostatique, nous permettant ainsi de déterminer le potentiel associé à une distribution de charges électriques. En effet le potentiel V vérifie:
$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = \frac{-\rho}{\epsilon_0}$

2.2 Espace d'étude

Notre étude se limite à un espace bi-dimensionnel définit par:
$x_i = x_0 + i\Delta x \qquad 0 \leq i \leq I$
$y_i = y_0 + j\Delta y \qquad 0 \leq j \leq J$
I et J sont fixés selon le problème traité.
Dans le contexte de ce problème, l'objectif est de résoudre l'équation associée au modèle du condensateur plan. Pour ce faire, il est nécessaire de spécifier les conditions qui faciliteront la création du maillage.

Cela implique la délimitation du domaine qui englobe le condensateur, ainsi que la détermination de la position des deux plaques du condensateur :

Domaine du condensateur :

$$0 \leq x \leq 1 \quad \text{u.a}, \quad 0 \leq y \leq 1 \quad \text{u.a.}$$

Lignes de charge positive :

$$0.25 \leq x \leq 0.75 \quad \text{u.a}, \quad y = 0.4 \quad \text{u.a.}$$

Lignes de charge négative :

$$0.25 \leq x \leq 0.75 \quad \text{u.a}, \quad y = 0.6 \quad \text{u.a.}$$

On cherche donc une solution de l'équation sur le domaine carré $[0, 1] \times [0, 1]$. Ici, les conditions aux bords imposées sont de Dirichlet, c'est-à-dire que la fonction V est nulle aux bords du maillage. Ces conditions aux bords permettent d'assurer l'unicité de la solution.

Le maillage de l'espace ici est donné avec $I = J = 65$ points, donc l'espacement du maillage $\Delta x = \Delta y = 1/65$. On choisit l'origine du repère $(x_0, y_0) = (0, 0)$.

Voici le condensateur plan ainsi que son maillage et les bords:

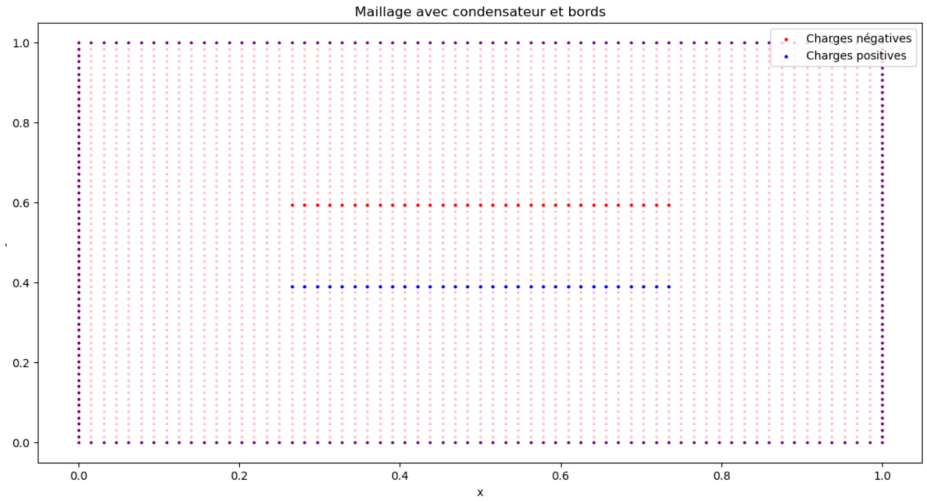

Figure 1: Maillage avec condensateur et bords

2.3 Mise en Place de la Résolution

2.3.1 Développement de Taylor

Afin de déterminer les valeurs discrètes de u à chaque point (i, j) du maillage, notées $u_{i,j}$, on discrétise le laplacien de l'équation de Poisson à l'aide d'un développement de Taylor-Young d'ordre 2. Les valeurs discrètes de u sont alors obtenues comme suit :

$$u(x + \Delta x, y) \approx u(x, y) + \Delta x \cdot \frac{\partial u}{\partial x}(x, y) + \frac{\Delta x^2}{2} \frac{\partial^2 u}{\partial x^2}(x, y) + O(\Delta x^3)$$

De même pour y, les dérivées partielles de u sont exprimées comme suit :

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x + \Delta x, y) + u(x - \Delta x, y) - 2u(x, y)}{\Delta x^2}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u(x, y + \Delta y) + u(x, y - \Delta y) - 2u(x, y)}{\Delta y^2}$$

Finalement, l'équation de Poisson s'exprime sous la forme:

$$\frac{1}{\Delta x^2}(u_{i+1,j} + u_{i-1,j} - 2u_{i,j}) + \frac{1}{\Delta y^2}(u_{i,j+1} + u_{i,j-1} - 2u_{i,j}) = \rho_{i,j}$$

2.3.2 Conditions aux limites

Nous avons des conditions aux limites de Dirichlet.

Sur les bords du domaine, ces conditions ont été imposées, signifiant que la valeur du potentiel est nulle aux bords. Mathématiquement, ces conditions se traduisent dans les équations déterminées ci-dessus comme suit :

- Sur le côté gauche ($i = 0$) :

$$\frac{1}{\Delta x^2}(u_{1,j} - 2u_{0,j}) + \frac{1}{\Delta y^2}(u_{0,j+1} + u_{0,j-1} - 2u_{0,j}) = \rho_{0,j}$$

- Sur le côté droit ($i = I - 1 = N - 1$) :

$$\frac{1}{\Delta x^2}(u_{I-2,j} - 2u_{I-1,j}) + \frac{1}{\Delta y^2}(u_{I-1,j+1} + u_{I-1,j-1} - 2u_{I-1,j}) = \rho_{I-1,j}$$

- En bas ($j = 0$) :

$$\frac{1}{\Delta x^2}(u_{i,1} - 2u_{i,0}) + \frac{1}{\Delta y^2}(u_{i,0+1} + u_{i,0-1} - 2u_{i,0}) = \rho_{i,0}$$

• En haut ($j = J$) :

$$\frac{1}{\Delta x^2}(u_{i,J-1} - 2u_{i,J}) + \frac{1}{\Delta y^2}(u_{i,J+1} + u_{i,J-1} - 2u_{i,J}) = \rho_{i,J}$$

2.3.3 Possibilité en 1D

Afin d'obtenir la solution à chaque point du maillage, nous introduisons un vecteur d'indices k, où les coordonnées représentent les points du maillage. Formellement, cela se traduit par les équations suivantes :

$$\begin{cases} 0 \leq i < I \\ 0 \leq j < J \end{cases} \implies k = iJ + j, \quad 0 \leq k < IJ$$

La solution discrétisée, obtenue à partir du développement de Taylor, peut alors être formulée comme :

$$\frac{1}{\Delta x^2}(u_{k+J} + u_{k-J}) + \frac{1}{\Delta y^2}(u_{k+1} + u_{k-1}) - 2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right) u_k = \rho_k$$

Cette équation représente une combinaison linéaire des composantes d'un vecteur, que l'on peut exprimer sous forme matricielle :

$$Au = s$$

Ici, $u = (u_k)_{0 \leq k < IJ}$ est le vecteur inconnu et $s = (s_k)_{0 \leq k < IJ}$ est le vecteur source. La matrice A, de taille $IJ \times IJ$, représente la combinaison linéaire de u dans l'équation de Poisson discrétisée et réindexée en 1D.

Pour notre problème avec $I = J = 65$, la matrice A est de taille $65^4 = 17850625$ éléments. Malgré sa taille, seule une poignée d'éléments par ligne, correspondant aux coefficients de l'équation linéaire, sont non nuls, faisant de A une matrice creuse.

Conditions aux limites de Dirichlet

En 1D, sur les côtés gauche et droit, les conditions aux limites sont respectées en assignant des zéros, car la matrice des coefficients présente uniquement cinq diagonales non nulles : la diagonale principale, les diagonales $k + 1$, $k - 1$, $k + J$, et $k - J$.

Il reste à déterminer comment implémenter les conditions aux bords en haut et en bas. Pour le bas, dans la matrice 1D, le coefficient devant u_{k-1} doit être nul tous les J. Pour le haut, dans la matrice 1D, le coefficient devant u_{k+1} doit être nul à $J - 1$, puis $2J - 1$, et enfin $(I - 1) \times J - 1$.

2.4 Résolution

2.4.1 1ère méthode: Fonction de matrice creuse Sparse

On calcule des dérivées partielles de l'équation de Poisson en utilisant la méthode sparse. Initialement, nous travaillons sur une seule dimension avant de l'étendre à 2D sur la grille. Dans cette approche, étant donné que nous traitons des matrices majoritairement composées de zéros, nous utilisons la méthode sparse qui consiste à remplir les matrices uniquement aux emplacements où les éléments sont non nuls.

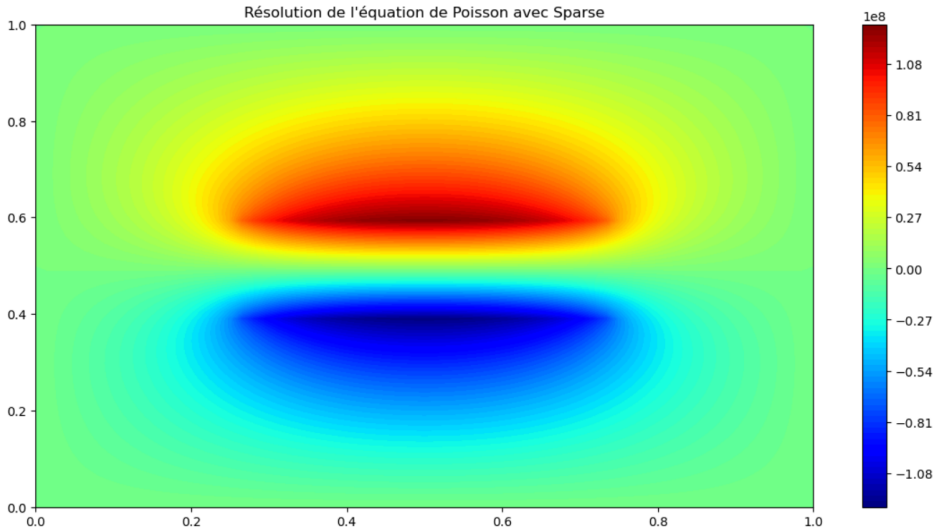

Figure 2: Modélisation par méthode Sparse

Il est important de noter que si la matrice n'est pas principalement constituée de zéros, cette méthode peut être inefficace, car elle remplit un par un les éléments de la matrice. Cependant, dans notre cas, où nous manipulons des matrices creuses, cette méthode offre une solution plus efficace.

2.4.2 2ème méthode: Fonction spsolve de scipy.sparse.linalg

On utilise ici une fonction python déjà réalisée nous permettant d'avoir très facilement notre modélisation à partir de notre schéma du condensateur: la fonction spsolve.

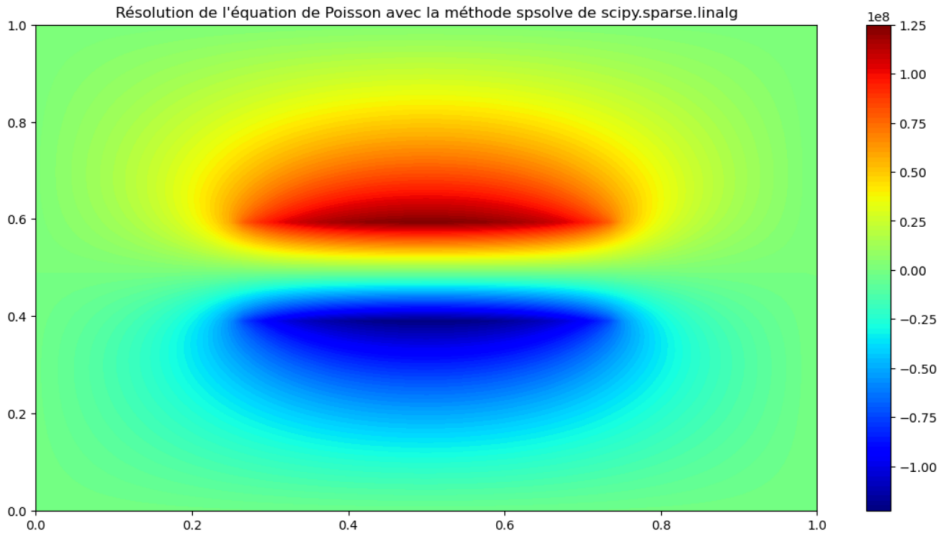

Figure 3: Modélisation par fonction spsolve

2.4.3 3ème méthode: le pivot de Gauss

On peut résoudre ce système par la méthode classique de résolution matricielle du Pivot de Gauss.

Cependant, cette méthode directe présente des inconvénients majeurs, notamment la nécessité d'un stockage en mémoire trop important et des durées d'exécution élevées lors de la manipulation de matrices de grande taille. En effet, le coût de l'algorithme du Pivot de Gauss est de l'ordre de N^3, ce qui peut devenir prohibitif pour de grandes valeurs de N. De plus, la méthode du Pivot de Gauss

est numériquement instable, accumulant les erreurs de calcul et conduisant parfois à des solutions numériques très éloignées de la solution exacte.

Étant donné que la plupart des coefficients de la matrice sont nuls, il est possible d'utiliser cette information pour minimiser le nombre de calculs et optimiser la résolution du système en termes de temps et de mémoire. C'est l'intérêt d'utiliser des méthodes alternatives, plus performantes pour des matrices creuses.

Dans notre cas, la matrice est de taille 65×65. Étant donné que la RAM de l'ordinateur n'est pas suffisamment élevée pour stocker une matrice de taille 4225×4225 nécessaire à l'inversion par Pivot de Gauss, nous utilisons le module *sparse* afin de pouvoir stocker en mémoire la matrice, qui est creuse. Ce module permet de ne conserver en mémoire que la valeur et la position des termes non nuls de la matrice.

2.4.4 4ème Méthode : Jacobi

La méthode de Jacobi est un algorithme itératif visant à trouver les solutions d'un système d'équations linéaires.

Considérons le système $Au = s$ défini précédemment, avec A une matrice carrée d'ordre N. On peut exprimer cette matrice comme la somme de trois matrices $A = D - L - U$, où D est une matrice diagonale, L est une matrice triangulaire inférieure, et U est une matrice triangulaire supérieure. Nous réécrivons notre système comme suit :

$$Au = s \quad \Rightarrow \quad Du - (L+U)u = s \quad \Rightarrow \quad u = D^{-1}(L+U)u + D^{-1}s$$

Nous définissons ensuite une suite de vecteurs (u^k) à partir d'un vecteur initial u^0 (généralement un vecteur nul) et de la formule de récurrence :

$$u^{k+1} = D^{-1}(L+U)u^k + D^{-1}s$$

Nous cherchons une suite (u^k) convergente vers une solution de notre système $Au = s$. Sans hypothèses préalables, cette convergence n'est pas garantie. Cependant, le théorème énonce que si A est une matrice à diagonale dominante, alors pour tout u^0, la suite (u^k) converge vers l'unique solution de $Au = s$. Dans notre contexte électrostatique, avec des conditions limites précises et une matrice creuse, ce théorème s'applique.

La relation de récurrence peut être réécrite en termes d'éléments matriciels plutôt qu'avec les matrices elles-mêmes, simplifiant ainsi l'implémentation :

$$u_i^{k+1} = \frac{1}{a_{i,i}} \left(s_i - \sum_{j=1}^{i-1} a_{i,j} u_j^k - \sum_{j=i+1}^{n} a_{i,j} u_j^k \right)$$

où $a_{i,j}$ sont les coefficients de la matrice creuse A, s_i et u_i sont les composantes respectives du vecteur source et du vecteur inconnu. L'indice k indique la k-ième itération.

Dans cet algorithme, le calcul de chaque u_i^{k+1} nécessite la connaissance de tous les u_j^k sauf u_i^k. Deux vecteurs de taille N sont donc nécessaires pour le stockage.

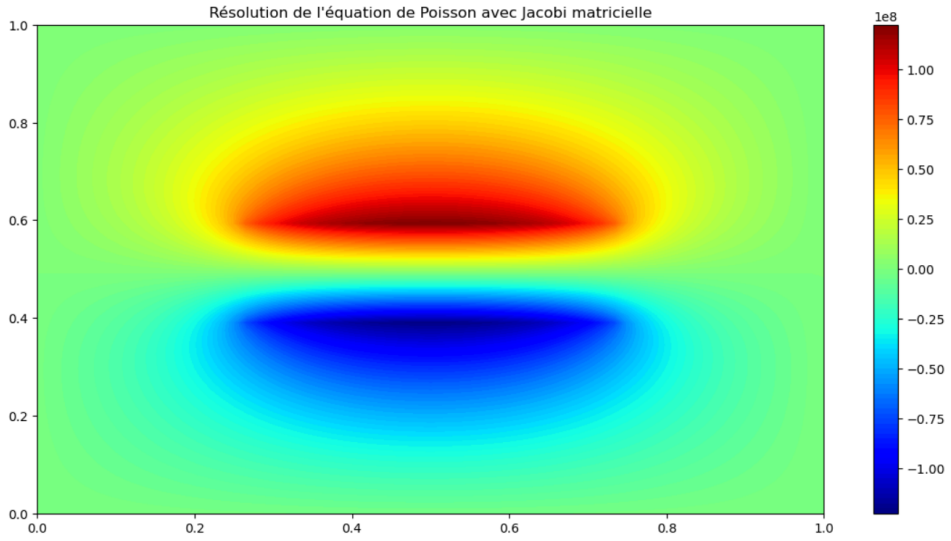

Figure 4: Modélisation par la méthode Jacobi

2.4.5 5ème Méthode : Gauss-Seidel

La méthode de Gauss-Seidel peut être vue comme une amélioration de la méthode de Jacobi. La formule de récurrence pour cette méthode s'exprime ainsi :

$$u_i^{k+1} = \frac{1}{a_{i,i}} \left(s_i - \sum_{j=1}^{i-1} a_{i,j} u_j^{k+1} - \sum_{j=i+1}^{n} a_{i,j} u_j^k \right)$$

On remarque que, pour le calcul de u_i^{k+1}, les valeurs déjà calculées de u_j^{k+1} pour $j < i$ sont utilisées. Ainsi, un seul vecteur de stockage est nécessaire, offrant un avantage en termes de mémoire par rapport à la méthode de Jacobi et une meilleure complexité pour des grandes valeurs de N.

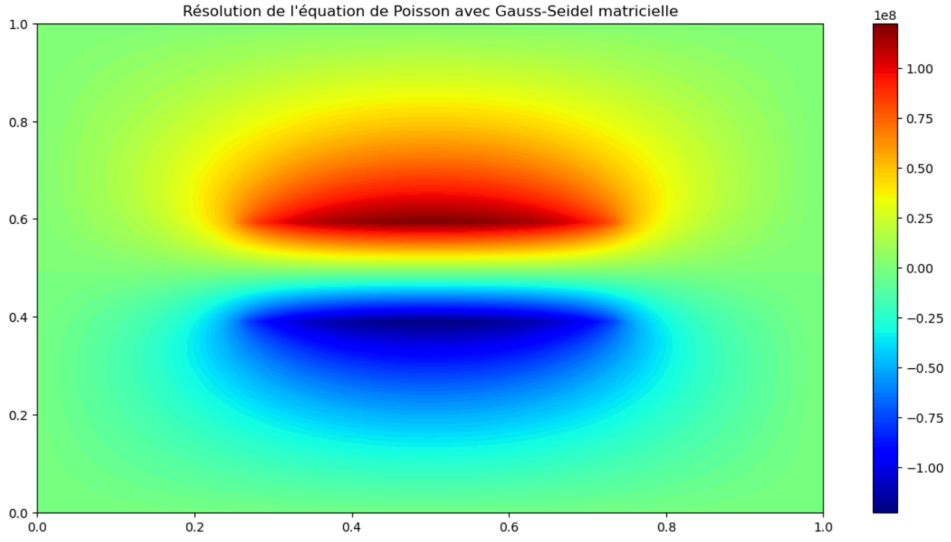

Figure 5: Modélisation par la méthode Gauss-Seidel

2.4.6 6ème Méthode : Accélération par sur-Relaxation

L'utilisation de l'accélération par sur-relaxation (SOR) vise à améliorer la convergence et la complexité de l'algorithme de Gauss-Seidel, initialement en $O(N^2)$ où N^2 est le nombre de points du maillage.

La méthode SOR est itérative, ajustant les valeurs de la solution obtenue par Gauss-Seidel avec la formule :

$$u_{i,k+1} = (1-\omega)u_{i,k} + \frac{\omega}{a_{ii}}\left(s_i - \sum_{j<i} a_{ij}u_{j,k+1} - \sum_{j>i} a_{ij}u_{j,k}\right)$$

Ainsi, des corrections successives, pondérées par un facteur de relaxation ω, sont apportées aux valeurs de u. Pour assurer la convergence, ω doit être dans l'intervalle [0 ; 2]. L'accélération est obtenue lorsque $1 < \omega < 2$ (la méthode de Gauss-Seidel correspond à $\omega = 1$).

Dans notre contexte électrostatique, avec des conditions de Dirichlet aux bords et une matrice creuse, le choix optimal de ω est connu : $\omega_{opt} = \frac{2}{1+\sin(\frac{\pi}{N})}$ avec $N = 65$.

Figure 6: Modélisation par la méthode de sur-relaxation

3 Comparaison des durées de modélisation

Les résultats de la résolution matricielle ainsi que des méthodes itératives de Gauss-Seidel et Jacobi sont obtenus. Pour Jacobi, le nombre d'itérations avant convergence est de 8793, tandis que pour Gauss-Seidel, il est de 4700, démontrant une convergence plus efficace de cette dernière. Cependant, les temps d'exécution sont relativement longs.

Afin d'optimiser mémoire et temps, une implémentation coefficient par coefficient est proposée, profitant de la structure creuse de la matrice des coefficients de Poisson. Cela réduit l'utilisation de mémoire, mais les temps d'exécution sont plus longs que pour la méthode matricielle.

L'accélération par sur-relaxation est ensuite appliquée à la méthode de Gauss-Seidel avec $\omega = 1.9$, améliorant les temps d'exécution.

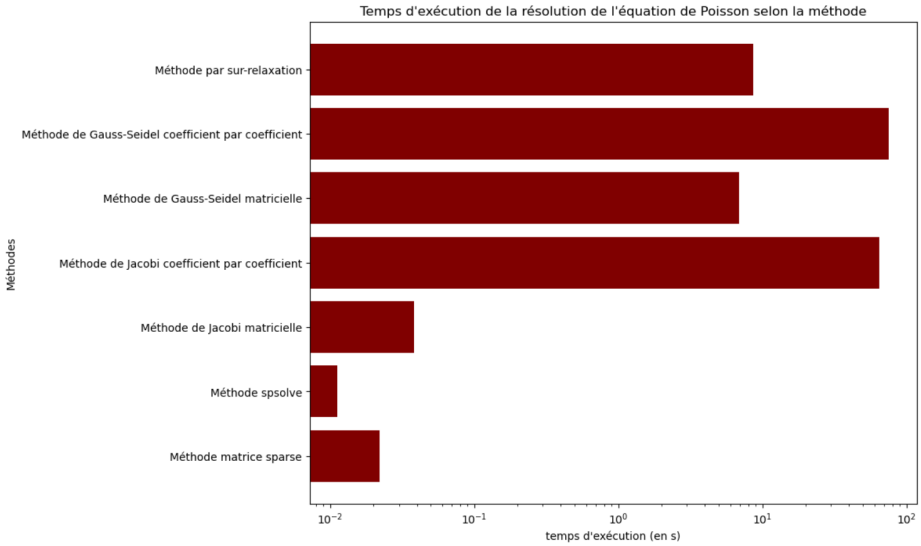

Figure 7: Temps de modélisation des différentes méthodes

En analysant les durées d'exécution, on constate que les méthodes natives d'inversion sont rapides, mais leur complexité en mémoire est inconnue. Les méthodes matricielles sont plus rapides que les méthodes coefficient par coefficient, avec néanmoins la méthode Jacobi matricielle surpassant la méthode Gauss-Seidel matricielle. Gauss-Seidel est plus lent en raison de l'inversion d'une somme de matrices de grande taille. Malgré cela, la convergence est plus rapide pour Gauss-Seidel, nécessitant la moitié des itérations de Jacobi. Les méthodes coefficient par coefficient sont plus lentes, en particulier avec 10000 itérations, mais Gauss-Seidel pourrait être accéléré significativement avec la sur-relaxation.