Problème avec grille structurée

Zhang minghe, Farid Diyae-eddine

1 Description du problème

Ici on s'intéresse à approcher la solution de l'équation de Poisson 2D:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 y}{\partial y^2} = f(x,y) \quad pour \quad (x,y) \in \Omega =]0,a[\times]0,b[\\ u(x,0) = U_0, \quad u(x,b) = U_0, \quad u(a,y) = U_0, \quad u(0,y) = U_0(1+\alpha V(y)); \end{cases}$$

avec $V(y) = 1 - cos(2\pi y/b)$
Pour résoudre ce problème, on considère le schéma de différences finies

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

Pour $i = 0...N_x$, $j = 0...N_y$
on a bien un système linéaire $Au = b$, avec u est la solution, $b = f_{i,j}$ et

$$A = \begin{pmatrix} T & I & & \\ I & \ddots & \ddots & \\ & \ddots & \ddots & I \\ & & I & T \end{pmatrix}, T = \begin{pmatrix} \frac{-2}{\Delta x^2} + \frac{-2}{\Delta y^2} & \frac{1}{\Delta x^2} & & \\ \frac{1}{\Delta x^2} & \ddots & \ddots & \\ & \ddots & \ddots & \frac{1}{\Delta x^2} \\ & & \frac{1}{\Delta x^2} & \frac{-2}{\Delta x^2} + \frac{-2}{\Delta y^2} \end{pmatrix}, I = \begin{pmatrix} \frac{1}{\Delta y^2} & & \\ & \ddots & \\ & & \frac{1}{\Delta y^2} \end{pmatrix}$$

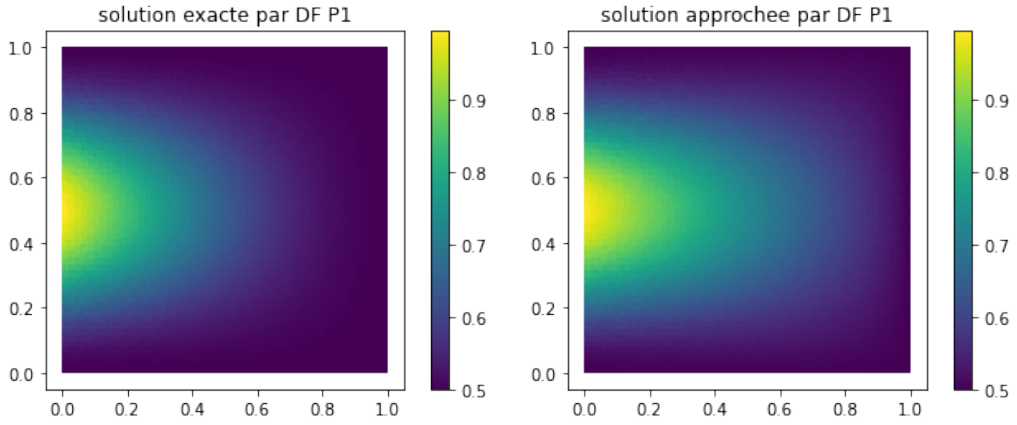
On se propose de résoudre le problème ci-dessus par une méthode de Jacobi, puis par une méthode de Gauss-Seidel.
Dans notre cas, $U_0 = 0.5, \alpha = 0.5, a = b = 1, N_x = N_y = 50$, le nombre d'itération $L = 10000$ la fonction $u(x,y) = U_0(1 + \alpha(1 - sin(\frac{\pi}{2}x))(1 - cos(2\pi y)))$

2 Méthode de Jacobi

Pour trouver la solution du système linéaire en utilisant cette méthode, on peut écrire $A = D - L - U$, avec D est une matrice diagonale, L une matrice triangulaire inférieur à diagonale nulle et U une matrice triangulaire supérieur à diagonale nulle. L'algorithme d'itération s'écrit

$$x_{k+1} = D^{-1}(L+U)x_k + D^{-1}b$$

D'après la méthode "5-points stencil", on peut obtenir la solution. Les visualisations graphiques et la performance sont ci-dessous:

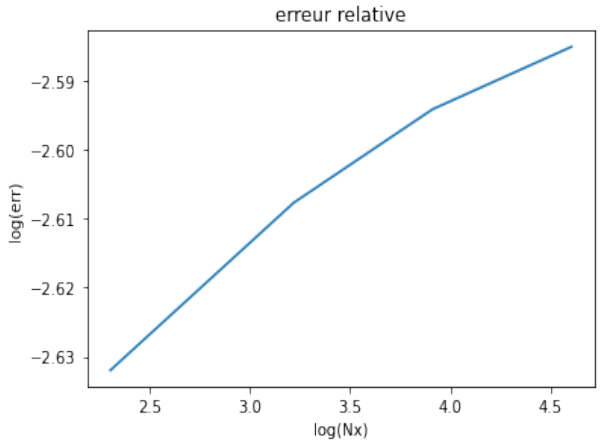


Runtime : 0.215494s Absolute error: 2.33453

Maintenant, on s'intéresse à vérifier la convergence numérique.

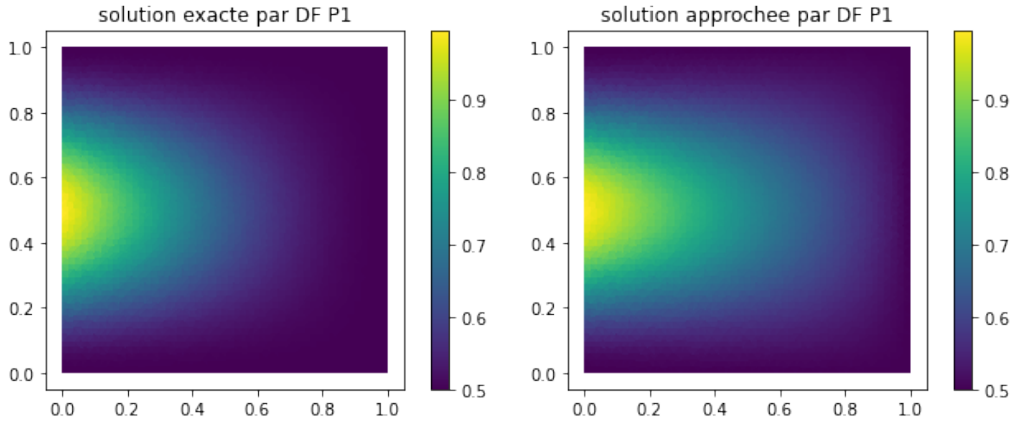
N_x	N_y	L	Time(seconde)	Erreur absolu	Erreur relatif
50	50	1000	0.022387	4.3866	0.14038
50	50	10000	0.222705	2.33453	0.074711
10	10	100000	0.092463	0.516308	0.0719362
25	25	100000	0.553274	1.1943	0.0737058
50	50	100000	2.16429	2.33453	0.074711
100	100	100000	8.76604	4.61908	0.07539

On peut voir que quand la taille de maillage augmente, l'erreur augmente un peu. Représentation graphique:



2.1 Parallélisation

Dans cette partie, on a divisé notre opération par 4, avec les même données que la partie précédente, on obtient




```
[myRank=3] 40 50 11
[myRank=2] 27 39 13
[myRank=0] 1 13 13
[myRank=1] 14 26 13
Runtime: 0.0761782s
Absolute error: 0.70818
```

L'idée de parallélisation c'est communiquer entre la ligne en haut et la ligne en bas, pour ce faire, il suffit de changer le longueur du message (ici c'est $N_y + 2$) et l'indice du vecteur.
On peut voir que la parallélisation marche bien, maintenant on s'intéresse à la performance

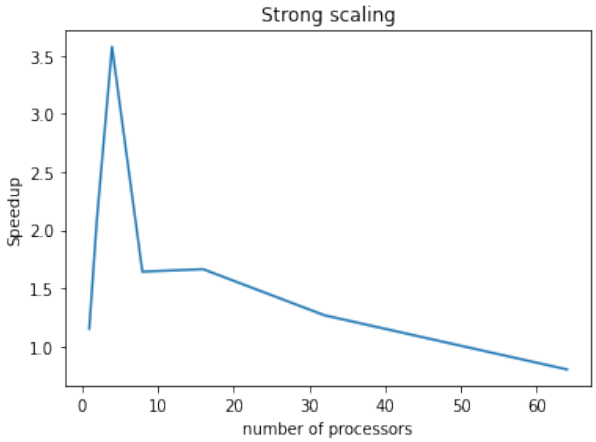
N_x	N_y	L	Time(seconde)	Time(sans parallélisation)
50	50	1000	0.0184731	0.022387
50	50	10000	0.0918667	0.222705
10	10	100000	0.113209	0.092463
25	25	100000	0.255461	0.553274
50	50	100000	0.65545	2.16429
100	100	100000	2.29226	8.76604

2.2 scalabilité

On fixe le nombre de itération $L = 10000$, et $N_x = N_y = 100$, la relation entre le Speedup S et le nombre de processus P ci-dessous:

S	1.150745	2.087268	3.577992	1.6417804	1.663498	1.266633	0.800272
P	1	2	4	8	16	32	64

Avec $T_{sequentiel} = 0.885317$
Représentation graphique:



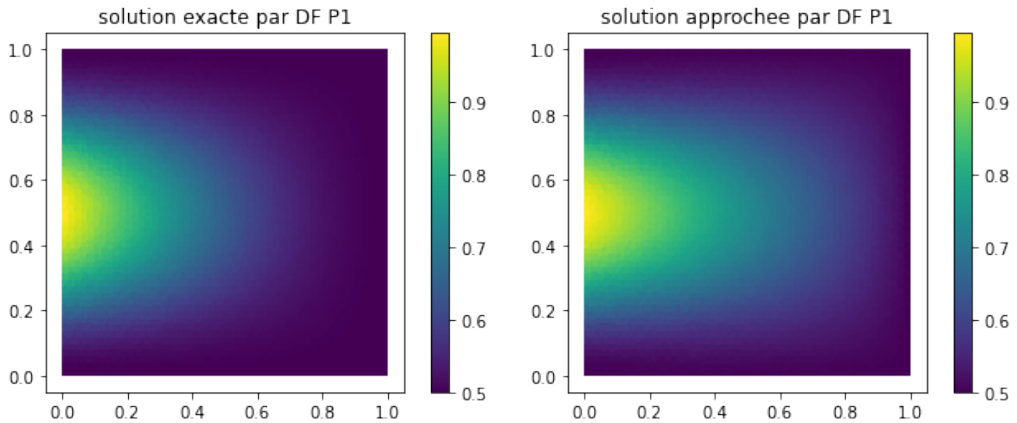
On peut voir que la courbe décroit quand le nombre de processus est supérieur à 4, parceque dans la machine d'ENSTA, il n'y a que 4 coeurs.

3 Méthode de Gauss-Seidel

Pour utiliser cette méthode, on fait la décomposition de la matrice A comme précédent, $A = D - L - U$, avec D, L et U la même forme. L'algorithme d'itération s'écrit

$$x_{k+1} = (D - L)^{-1} U x_k + (D - L)^{-1} b$$

Après coder cet algorithme en utilisant les mêmes parametres, avec la méthode red-black" coloring, on obtient



Runtime : 0.21326s
Absolute error: 2.33453

Si on compare la performance entre la méthode de jacobi et la méthode de gauss-seidel, on aura

N_x	N_y	L	Time(gauss-seidel)	Time(jacobi)
50	50	1000	0.04233	0.022387
50	50	10000	0.422929	0.222705
10	10	100000	0.169699	0.092463
25	25	100000	1.03079	0.553274
50	50	100000	4.20845	2.16429
100	100	100000	16.548	8.76604

On peut voir que la vitesse du code de la méthode de gauss-seidel est presque 2 fois moins rapide que celui de la méthode jacobi.

4 Parallélisation du code Jacobi en utilisant le cluster Cholesky

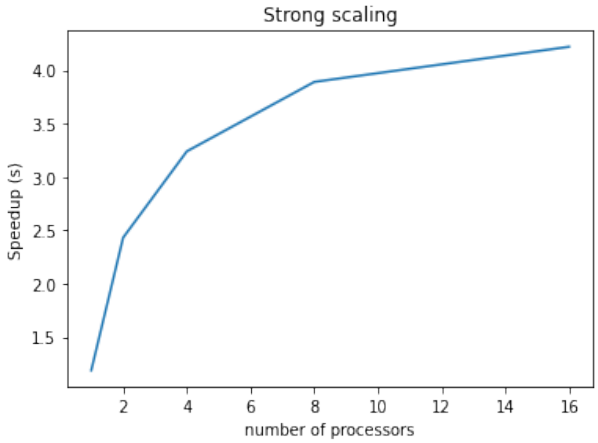
Dans cette partie, on va étudier la performance parallèle du code "Jacobi".

4.1 Scalabilité forte

On fixe le nombre d'itération $L = 10000$, et $N_x = N_y = 100$, la relation entre le Speedup S_p et le nombre de processus P est:

S_p	1	1.951954	3.2396008	3.889991	4.219391
P	1	2	4	8	16

Visualisation graphique:

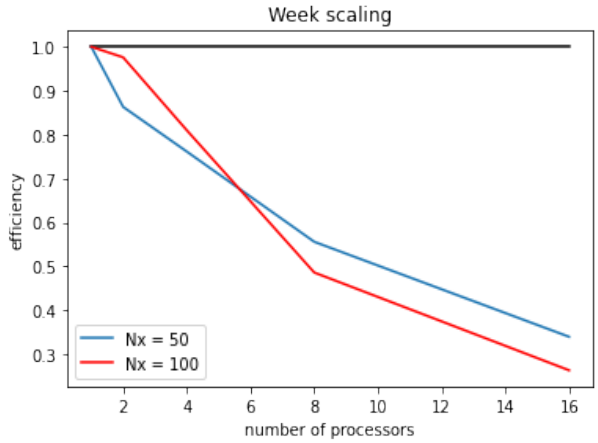


4.2 Scalabilité faible

La relation entre l'efficacité E et le nombre de processus P est:

	P	1	2	4	8	16
$N_x = 50$	E	1	0.8632	0.76142	0.556171	0.33994
	S	1	1.72641	3.0457	4.44937	5.439117
$N_x = 100$	E	1	0.97597	0.8099	0.48624	0.26371
	S	1	1.951954	3.2396008	3.889991	4.219391

Visualisation graphique:



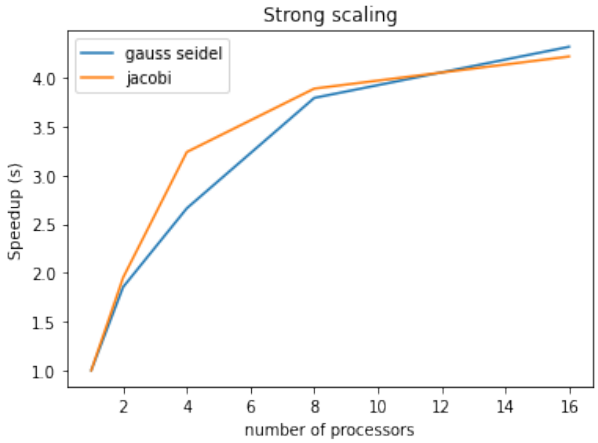
5 Parallélisation du code "Gauss-seidel"

L'idée de la parallélisation de cette méthode est presque la même que celle de la méthode de jacobi

5.1 Scalabilité forte :

S_{Gauss}	1	1.854367	2.662356	3.794428	4.319095
S_{jacobi}	1	1.951954	3.2396008	3.889991	4.219391
P	1	2	4	8	16

Visualisation graphique:



5.2 Scalabilité faible :

E_{Gauss}	1	0.927183	0.665589	0.4743035	0.2699434
E_{jacobi}	1	0.97597	0.8099	0.48624	0.26371
P	1	2	4	8	16

Visualisation graphique: