Rapport intermédiaire - projet 1

Paul Michel, Clément Baillet

7 Octobre 2025

1 Méthode de Jacobi

Dans toute la suite du rapport, les exemples sont réalisés pour les données suivantes : $U_0 = 1$, $\Omega =]0,1[\times]0,1[$, $\alpha = 0,5$; et on fixe un nombre maximal d'itérations à 10100 ainsi qu'une tolérance en précision de 10^{-6}.
Pour un nombre de points intérieurs $N = 50$ (selon x et selon y), on converge après 4235 itérations avec une erreur finale légèrement inférieure à 10^{-6} :

```
Iteration 4000, error = 1.56019e-06
Iteration 4100, error = 1.29041e-06
Iteration 4200, error = 1.06728e-06
Converged after 4235 iterations with error = 9.9867e-07
```

Figure 1: Exemple d'exécution du code séquentiel - méthode de Jacobi

2 Méthode de Gauss-Seidel

Pour le même jeu de données, on converge plus rapidement grâce à la technique de Gauss-Seidel :

```
Iteration 2000, error = 3.13557e-06
Iteration 2100, error = 2.14495e-06
Iteration 2200, error = 1.4673e-06
Iteration 2300, error = 1.00373e-06
Converged after 2301 iterations with error = 9.99931e-07
```

Figure 2: Exemple d'exécution du code séquentiel - méthode de Gauss-Seidel

3 Comparaison des deux approches

Dans la méthode de Jacobi, on met à jour u à la fin de chaque boucle, alors que pour Gauss-Seidel on le fait dans la boucle même. Cela implique l'utilisation de

2 "u", c'est-à-dire 2 grilles pour Jacobi, mais d'une seule pour Gauss-Seidel. Cette observation sur le plan logique des méthodes est confirmée par la différence du nombre d'itérations avant convergence : Gauss-Seidel est une méthode presque deux fois plus rapide que celle de Jacobi en ce sens d'après les figures 1 et 2. Plus encore, en comparant le temps d'exécution entre les deux approches, on remarque une fois de plus que celle de Gauss-Seidel est plus rapide.

4 Parallélisation du code : Jacobi

Dans cette section uniquement, les tests ont été effectués pour $N = 100$ points intérieurs (selon x et selon y), et exceptionnellement $\alpha = 0$, impliquant comme condition initiale $U = U_0 = 1$ sur tous les bords.
Le tableau ci-dessous résume les 4 tests réalisés pour cette section.

Nombre de process	4	30	60	100
Nombre d'itérations	10000	10000	6586	10000
erreur finale (x10^{-7})	30.9962	17.675	9.99853	30.9962

On observe une première croissance de la performance du code avec le nombre de processus (jusqu'à 60 process environ), avant que ce nombre ne soit trop élevée par rapport à la performance du réseau ENSTA, voire par rapport au nombre de tâches à effectuer. Toutefois, le code reste à améliorer puisque dans ce cas il reste tout juste plus performant que sa version séquentielle, ce qui ne devrait pas être le cas. En effet, pour $N = 100$ points intérieurs, on a les résultats suivants pour les algorithmes séquentiels :

Jacobi — Converged after 7613 iterations with error = 9.99568e-07
Gauss-Seidel — Converged after 4235 iterations with error = 9.9867e-07