



UNIVERSITÉ
CÔTE D'AZUR

Master 2 - Ingénierie Numérique

Master Ingénierie Mathématique

HPC

Compte Rendu - Projet

Étudiante :
Sarah ALI

Professeur :
Dr. Stéphane ABIDE

Janvier 2026

Problème

Dans ce projet on veut résoudre le problème 1D suivant :

$$-u'' + u = f \quad \text{sur le domaine } \Omega = [0, 2\pi].$$

Avec terme source :

$$f = 5 \cdot \sin(2x)$$

le Modèle discret (discrétisation au sens des différences finies):

$$-u''(x_i) = -\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2}$$

On obtient le problème suivant :

$$Au = f$$

telque :

$$A = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} + I$$

Méthode de résolution

Pour résoudre ce problème on implémente la méthode itérative de richardson :

$$u^{(k+1)} = u^{(k)} + \omega(f - Au^{(k)}) \quad \text{telque } \omega > 0.$$

Le terme $r^{(k)} = f - Au^{(k)}$ est le résidu.

L'implémentation numérique ne sera pas abordée dans ce rapport.

```
1 int main(int argc, char *argv[]) {
2     MPI_Init(&argc, &argv);
3     int size, rank;
4
5     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
6     MPI_Comm_size(MPI_COMM_WORLD, &size);
```

Initialisation de l'environnement MPI.

```
1 std::size_t N=32;
2 double L = 3* M_PI;
3 int maxIters = 20000;
4 double tol = 1e-6;
5 double h = L / (double)(size*N);
6 .
```

```

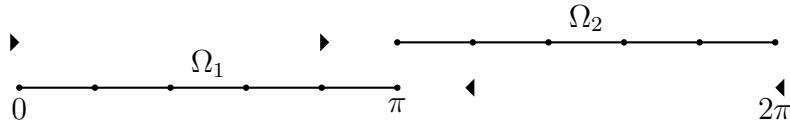
7   .
8   .
9     int other = 1 - rank;
10  .
11  .
12  .
13  .
14  for (int k = 0; k < maxIters; ++k) {
15
16    double uLghost = 0.0, uRghost = 0.0;
17
18    double send_left = u(0);
19    double send_right = u(N - 1);
20
21    MPI_Sendrecv(&send_left, 1, MPI_DOUBLE, other, 10,
22                  &uRghost, 1, MPI_DOUBLE, other, 10,
23                  MPI_COMM_WORLD, MPI_STATUS_IGNORE);
24
25    MPI_Sendrecv(&send_right, 1, MPI_DOUBLE, other, 20,
26                  &uLghost, 1, MPI_DOUBLE, other, 20,
27                  MPI_COMM_WORLD, MPI_STATUS_IGNORE);
28
29    b = A * u;
30    b(0) += -invh2*uLghost;
31    b(N-1) += -invh2*uRghost;

```

Dans cette partie on effectue les communications points à points, des valeurs aux bords des problèmes locaux, en utilisant MPI_sendrecv : si on prend la première communication , on affecte "send_left" à la valeur du bord gauche de notre vecteur de solution, qui est de taille "1", de type "double", le destinataire étant "other" , 1 si on est le rang 0 et 0 si on est le rang 1; et reçoie uRghost , de taille "1" et de type "double" depuis la source "other".

Pour clarifier, on se place sur le poit de bord droit du domaine ω_1 , il nous manque la valeur du voisin droite, mais aussi, le poit de bord gauche du domaine ω_2 a besoin de son voisin gauche, et donc le MPI_sendrecv il fait cet échange d'information comme suit :

"Moi le processus 0, j'ai besoin de l'information du processus 1, mais je doi échanger l'information qui lui manque de chez moi"



```

1   r = f - b;
2
3   double r2_local = r.squaredNorm();

```

```

4     double r2_other = 0.0;
5
6     MPI_Sendrecv(&r2_local, 1, MPI_DOUBLE, other, 30,
7                     &r2_other, 1, MPI_DOUBLE, other, 30,
8                     MPI_COMM_WORLD, MPI_STATUS_IGNORE);
9
10    double rnorm_global = std::sqrt(r2_local + r2_other);
11
12    double relres = rnorm_global / fnorm_global;
13
14    if (rank == 0 && (k % 2000 == 0)) {
15        std::cout << "iter " << k << " relres " << relres << "\n";
16    }
17    .
18    .
19    u.noalias() += omega * r;
20}
21.
22.
23.
24    MPI_Finalize();
25    return 0;
26}

```

Dans cette partie, on communique la norme du résidu local entre les processus, afin de pouvoir calculer une résidu locale, et arrêter les problème dès que ce résidu sera plus petit que la tolérance.

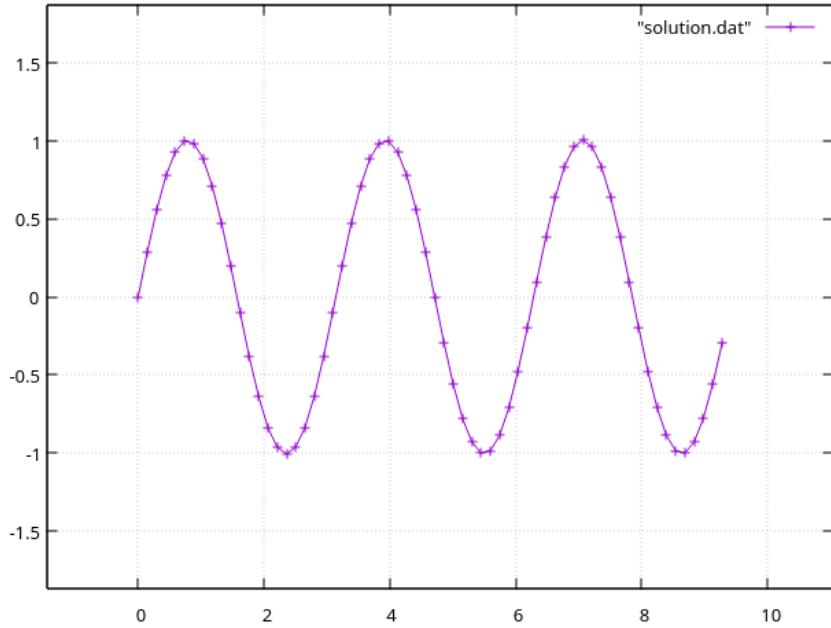


Figure 1: Solution over the domain $[0, 3\pi]$