TP mpi4py

Loïc Gouarin et Romaric David

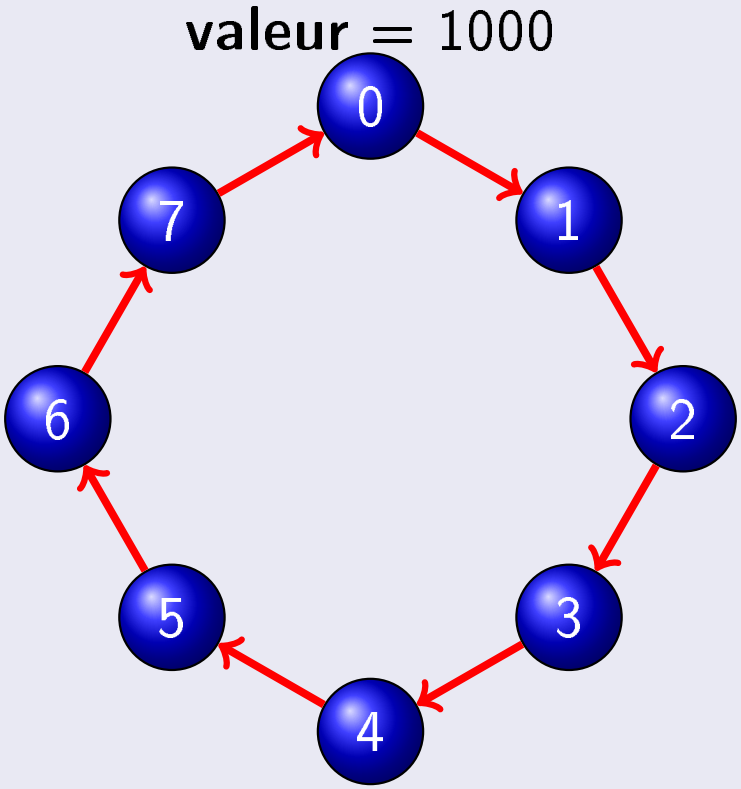
9 décembre 2010

Plan

Plan

Communication en anneau

Ecrire un script Python réalisant une communication en anneau. Le processus 0 a une valeur initialisée à 1000 et envoie à son voisin, le voisin rajoute 1 à cette valeur et envoie au suivant...



Plan

Le problème sur un domaine

On souhaite résoudre en parallèle le problème suivant

$$\begin{cases} \Delta u = 0 \quad \text{sur} \quad \Omega = [0,1] \times [0,1], \\ u = g \quad \text{sur} \quad \partial\Omega. \end{cases}$$

par un schéma aux différences finies à 5 points. On a donc le shéma interne suivant :

$$\frac{4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}}{h^2},$$

en supposant que $h = \Delta x = \Delta y$.

Le problème sur un domaine

On peut écrire le système linéaire de ce problème de la manière suivante :

$$Ax = b,$$

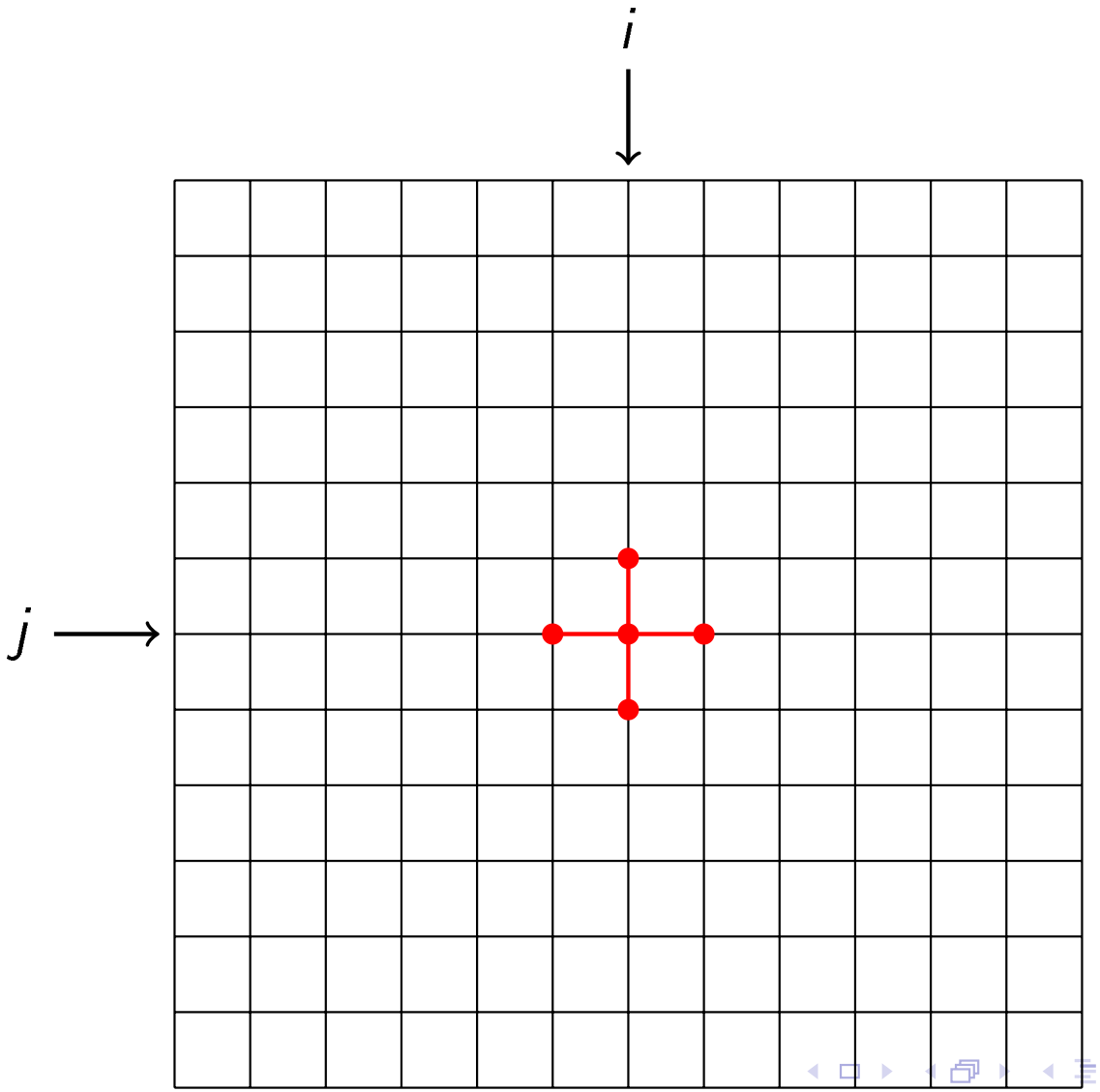
avec $A = D - E - F$, où

- D est la partie diagonale de A,
- E est la partie triangulaire inférieure de A,
- F est la partie triangulaire supérieure de A.

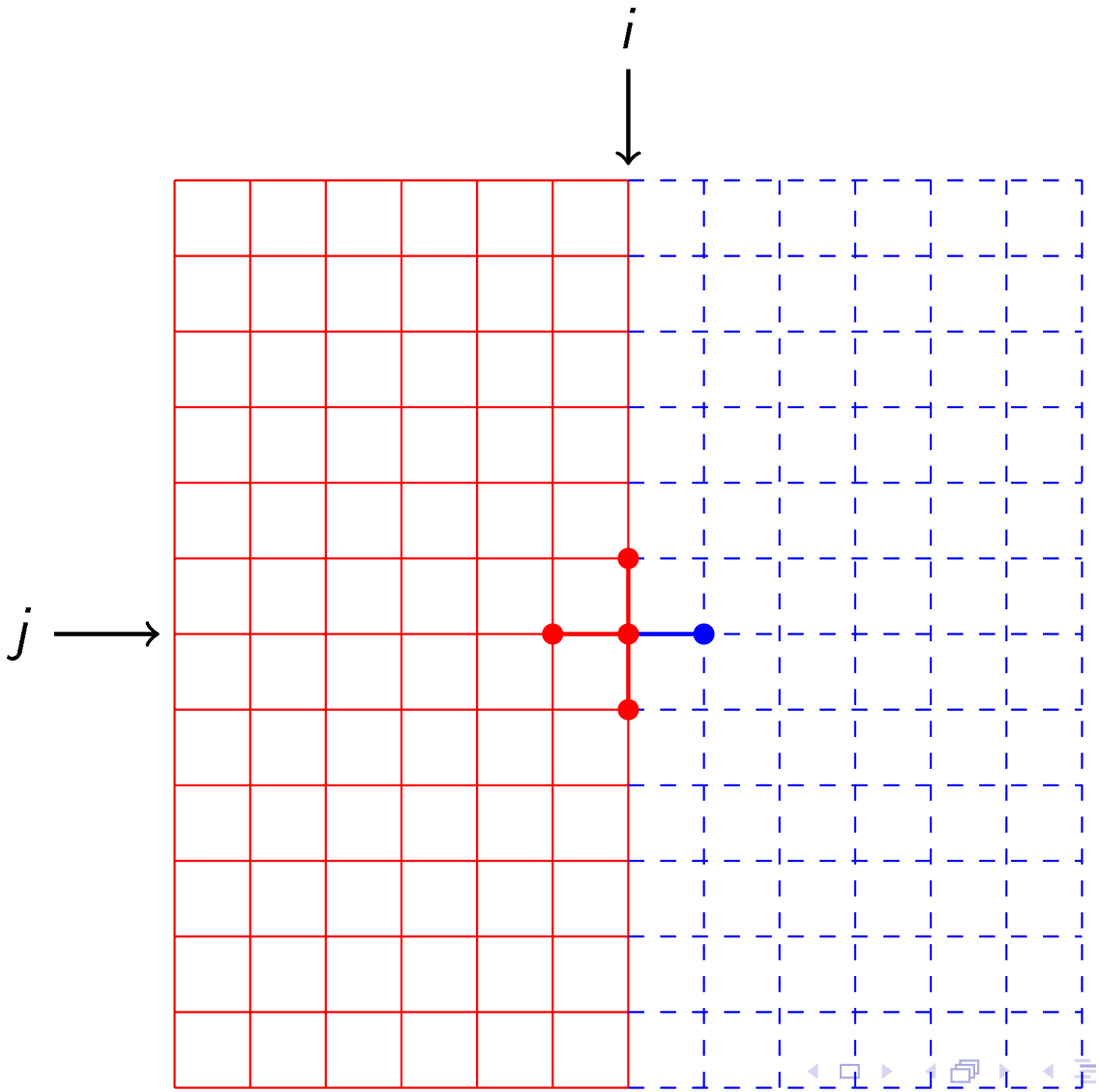
On peut donc utiliser une méthode de Jacobi pour résoudre le problème.

$$x^{k+1} = D^{-1}(E + F)x^k + D^{-1}b$$

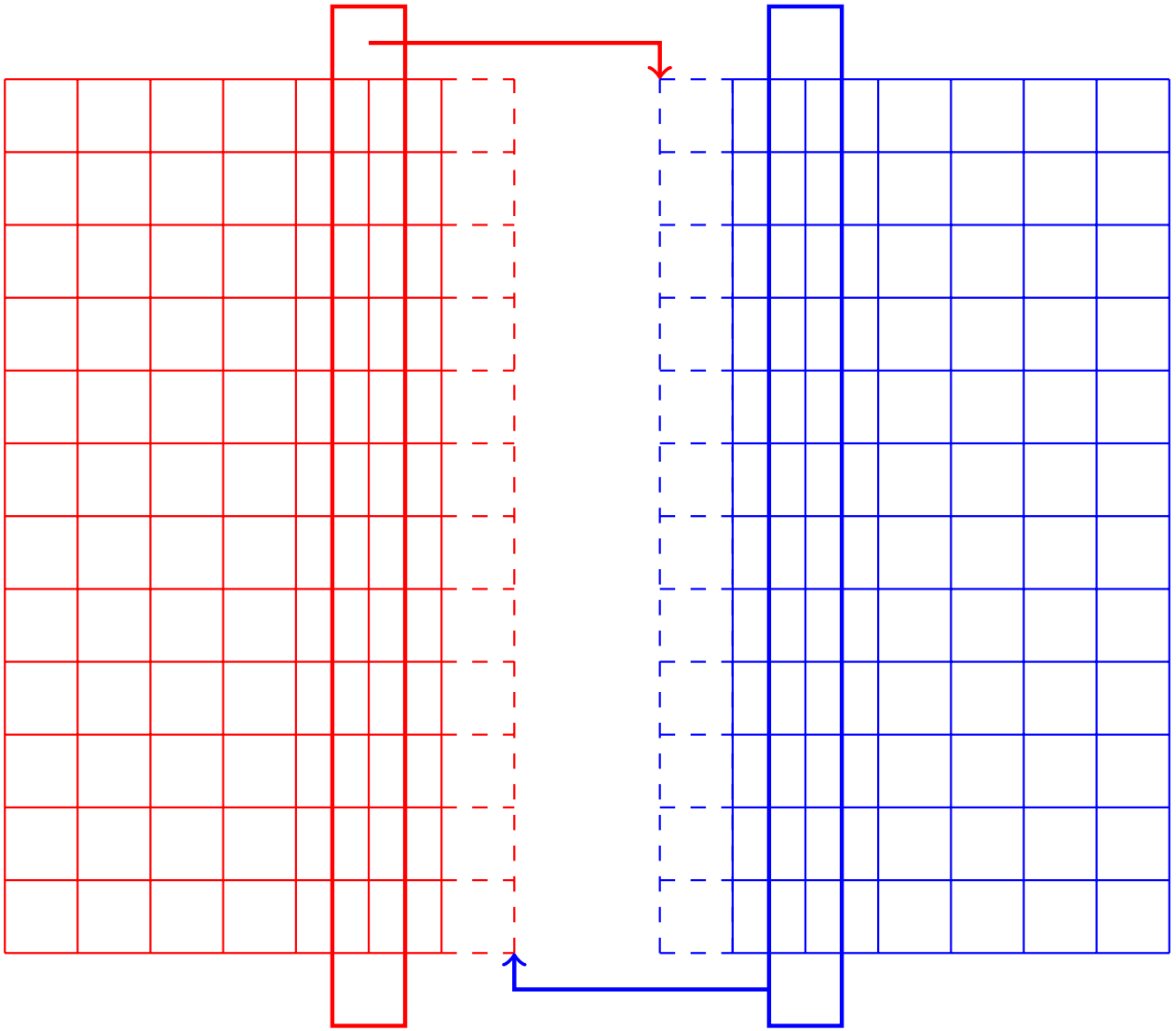
Parallélisation du système linéaire



Parallélisation du système linéaire



Parallélisation du système linéaire



Parallélisation du système linéaire

Algorithme de Schwarz classique

$$\begin{cases} \Delta u_1^{j+1} = 0 \quad \text{sur} \quad \Omega_1 = [0,.5] \times [0,1], \\ u_1^{j+1} = u_2^j \quad \text{sur} \quad \Gamma_{12}, \\ u_1^{j+1} = g \quad \text{sur} \quad \partial\Omega_1 \backslash \Gamma_{12}. \end{cases}$$

$$\begin{cases} \Delta u_2^{j+1} = 0 \quad \text{sur} \quad \Omega_2 = [.5,1.] \times [0,1], \\ u_2^{j+1} = u_1^j \quad \text{sur} \quad \Gamma_{12}, \\ u_2^{j+1} = g \quad \text{sur} \quad \partial\Omega_1 \backslash \Gamma_{12}. \end{cases}$$

où j représente une itération de l'algorithme et Γ_{12} représente l'interface entre les sous-domaines 1 et 2.

Parallélisation du système linéaire

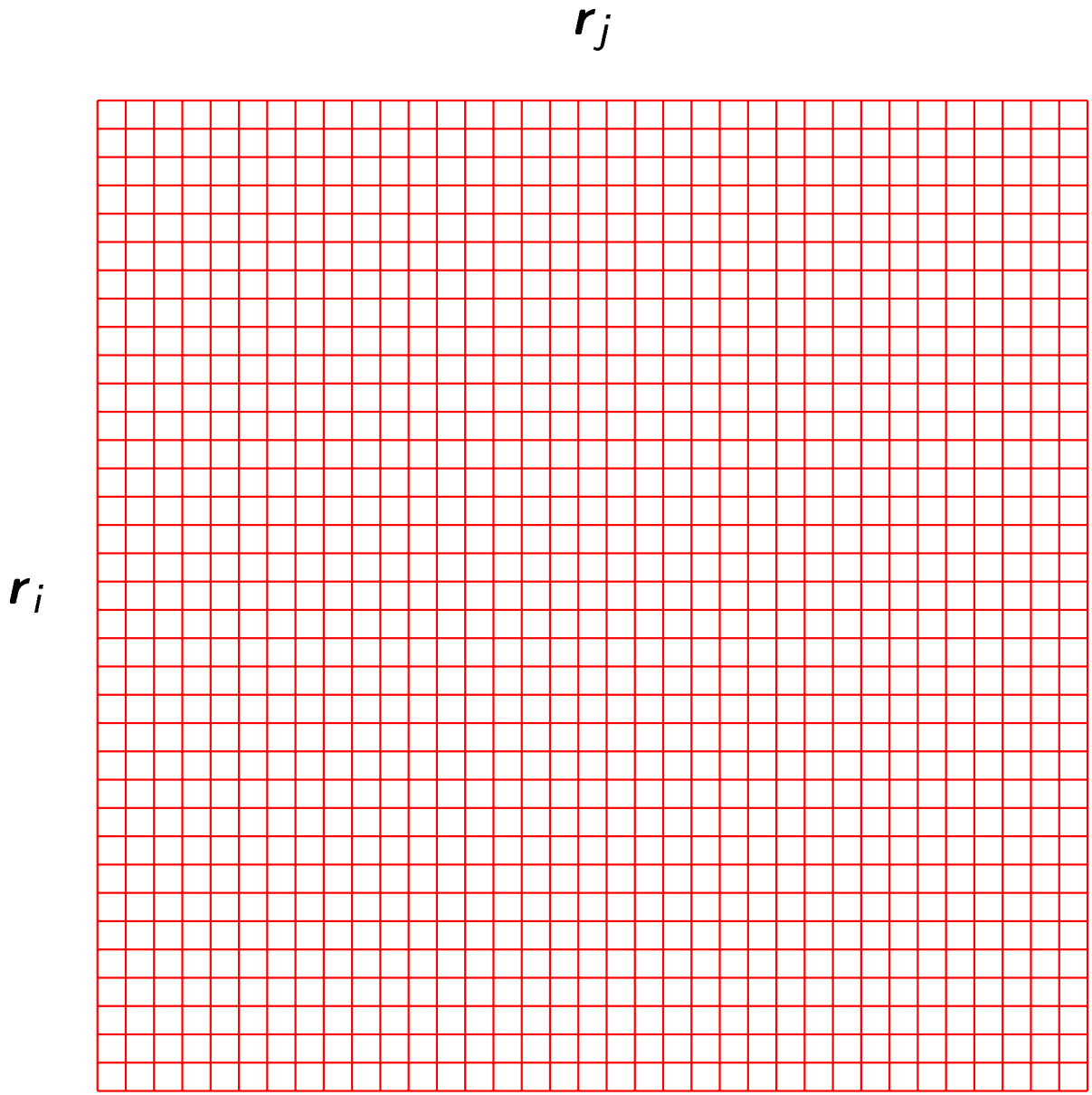
A faire

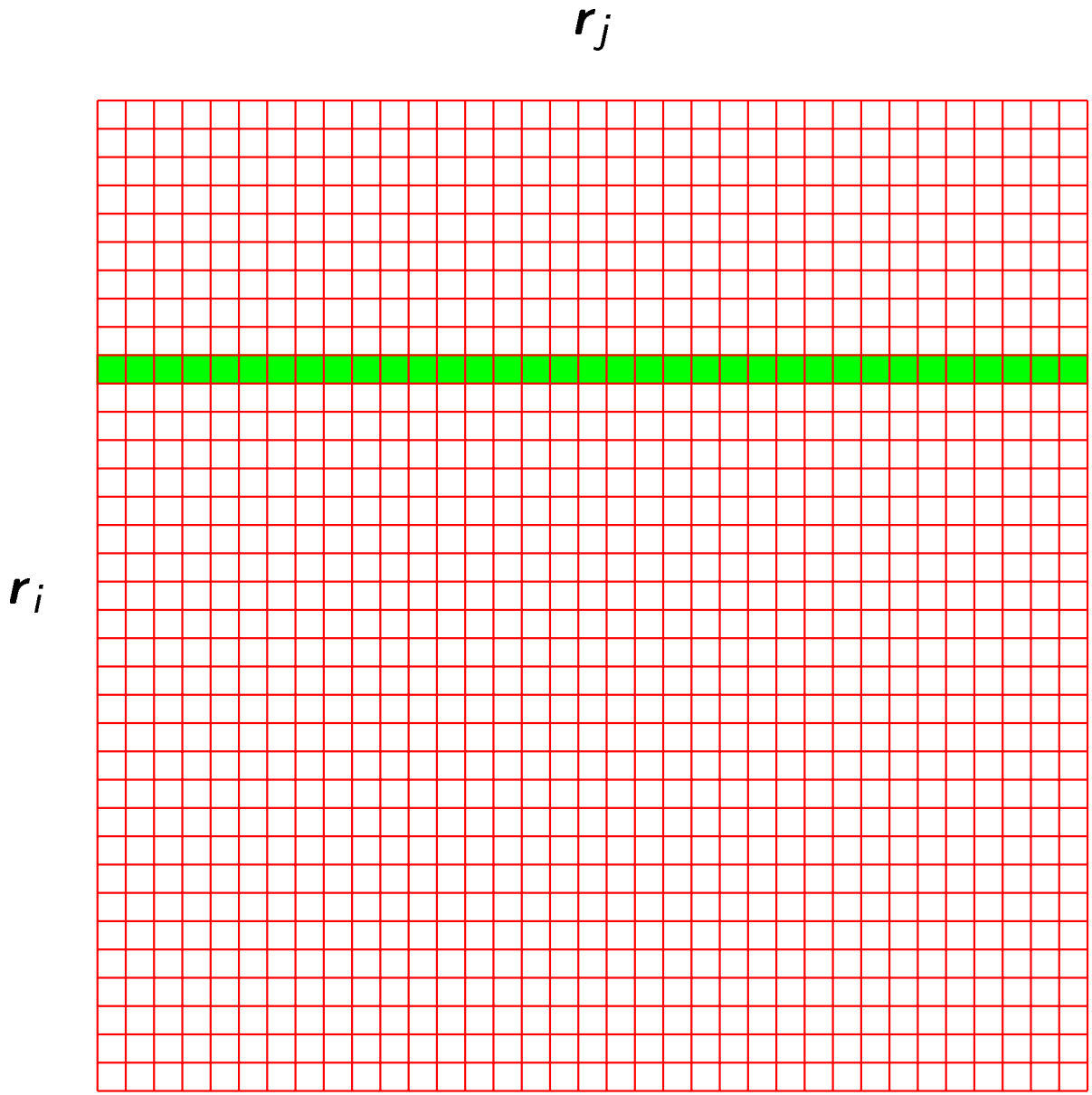
- Découper le domaine en 2.
- Chaque processus résout son système linéaire.
- Echanger les données aux bords pour réaliser l'itération suivante de la méthode de Jacobi.
- Calculer le nouveau résidu à l'aide d'un *Allreduce*.

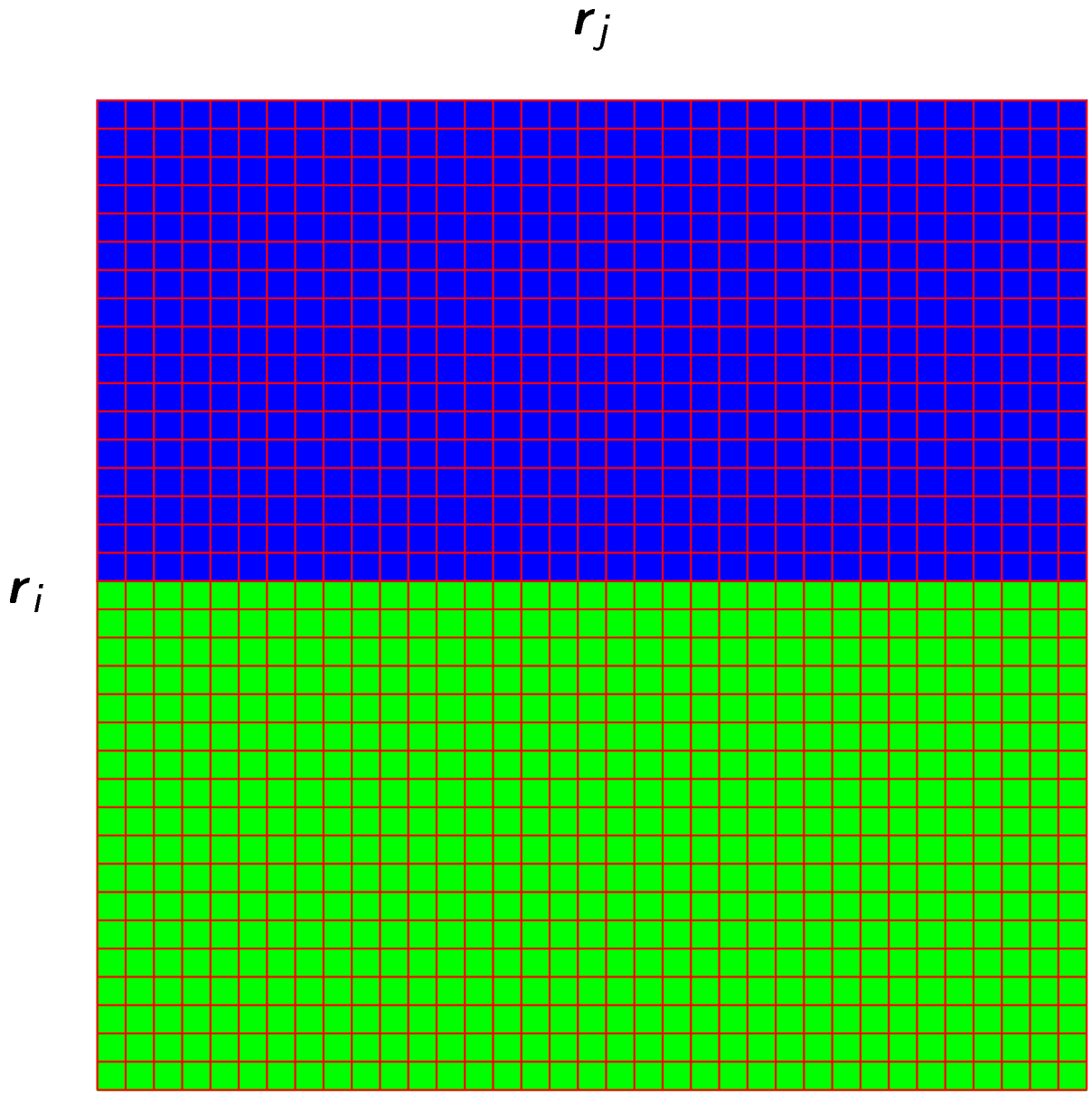
Plan

Calcul des forces

$$\boldsymbol{f}_i = -G \sum_{j \neq i} \frac{m_i m_j}{|\boldsymbol{r}_i - \boldsymbol{r}_j|^3} (\boldsymbol{r}_i - \boldsymbol{r}_j)$$


r_j
r_i


r_j
r_i


r_j
r_i

Parallélisation du système solaire

Initialisation, répartition

1. Le nombre de planètes traité par chaque processus correspond au nombre de planètes total divisé par le nombre de processus. Si le nombre de processus n'est pas un multiple du nombre de planètes, les planètes restantes seront traitées par le dernier processus.
2. Pour chaque processus, calculez l'indice de la 1ère et de la dernière planète traitée et diffusez le nombre d'éléments de chacun à tous les processus.

Parallélisation du système solaire

Calcul des forces, des vitesses et des positions

1. Lors du calcul des forces, chaque processus calcule les attractions exercées sur les planètes dont il a la charge.
2. Dans move, chaque processus calcule la nouvelle vitesse dont il a la charge.
3. On redistribue les nouvelles positions pour recalculer les nouvelles forces.
4. Et on itère en temps.