Chapitre **14**

Parallélisation des méthodes de Krylov

Les méthodes de Krylov requièrent trois types d'opération : produit matrice-vecteur, produit scalaire et combinaison linéaire de vecteurs. Pour paralléliser ces différentes opérations, il faut attribuer aux différents processus des morceaux des matrices et des vecteurs. Faire une partition de l'ensemble des équations revient à découper les vecteurs et la matrice par sous-ensembles de lignes, ce qui revient à un découpage en blocs de lignes de la matrice.
La parallélisation des combinaisons linéaires est triviale : chaque processus effectue le calcul sur les sous-vecteurs qui lui sont attribués. Pour les produits scalaires, chaque processus calcule la contribution de ses sous-vecteurs, l'assemblage des différentes contributions étant une opération de réduction standard.
Reste le plus important : le produit matrice-vecteur. C'est ce point qui est abordé en détails dans le chapitre, et plus particulièrement dans le cadre des méthodes de sous-structuration pour les matrices creuses.

14.1 PARALLÉLISATION DU PRODUIT MATRICE-VECTEUR PLEIN

La partition de l'ensemble des équations conduit à allouer à chaque processus le bloc de lignes correspondant au morceau des vecteurs qu'il traite. Dans la figure 4.1, ces termes apparaissent en noir. Cependant, le processus qui va réaliser le produit matrice-vecteur $y = Ax$ pour ce bloc de lignes ne possède a priori que les termes

correspondants du vecteur x, eux aussi en noir dans la figure 4.1. Afin d'effectuer ce produit, le processus a besoin de tous termes du vecteurs x. Il va donc devoir récupérer les termes qui lui manquent, en gris foncé sur la figure 4.1.

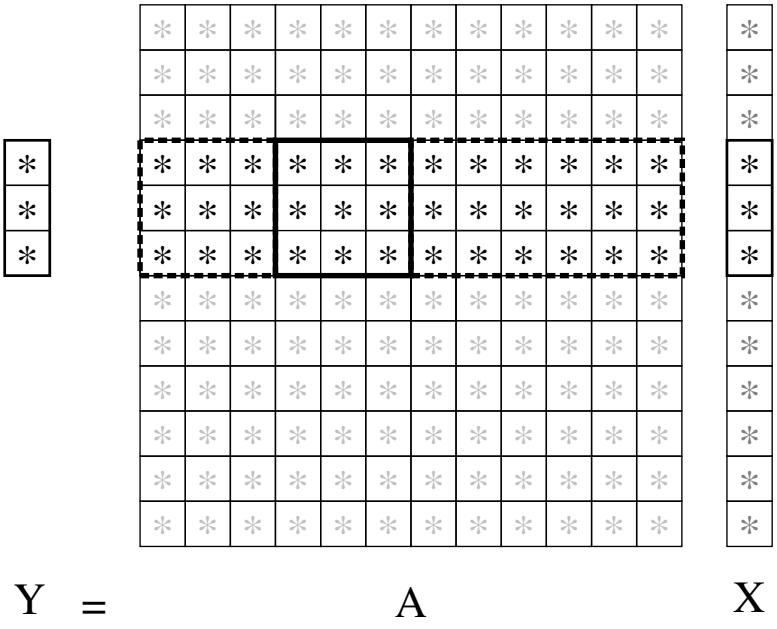

FIGURE 4.1. Produit par un bloc de lignes plein.

Comme il en est de même pour tous les processus, il va donc falloir reconstituer le vecteur x complet partout, à l'aide des morceaux répartis. Il s'agit d'un procédé d'échange collectif classique, où chacun est à la fois émetteur et récepteur, et dont la syntaxe MPI est :
MPI_Allgather(tableau_local,nombre_de_données_émises,type_de_donnée_émise, tableau_complet,nombre_de_données_reçues,type_de_donnée_reçue,communicateur)
Les paramètres
nombre_de_données_émises,type_de_donnée_émise et
nombre_de_données_reçues,type_de_donnée_reçue sont redondants : ils décrivent respectivement la taille du vecteur x local dans chaque processus et son type, c'est-à-dire ce que le processus émet, et la taille et le type des données reçues par chaque processus en provenance de chacun des autres. Le *tableau_complet* contient, dans chaque processus, les différents tableaux locaux, de même taille, rangés les uns derrière les autres en fonction du numéro de processus. Cela suppose donc que le taille du vecteur x local est uniforme, et que le premier processus traite le premier

sous-vecteur, le second processus, le second sous-vecteur, et ainsi de suite, ce qui est évidemment la situation la plus simple.

MPI permet aussi de réaliser le même type d'opération de reconstitution d'un tableau global avec des vecteurs locaux de longueurs variables rangés en réception dans un ordre différent de celui des numéros de processus. La fonction s'appelle alors *MPI_Allgatherv*, le *v* signifiant variable, et le paramètre entier *nombre_de_données_reçues* est alors remplacé par deux tableaux d'entiers de dimension égale au nombre de processus,
nombre_de_données_reçues,décalage_des_données_reçues,
décrivant respectivement, le nombre, variable, de données reçues de chacun des processus, et la position dans laquelle elles sont rangées dans le tableau de réception.

14.2 PARALLÉLISATION DU PRODUIT MATRICE-VECTEUR CREUX PAR ENSEMBLE DE POINTS

14.2.1. Produit matrice-vecteur

Si on applique le même procédé pour une matrice creuse, un processus traite un ensemble de lignes de la matrice et doit réaliser le produit par son bloc rectangulaire creux. Ne disposant a priori que des sous-vecteurs correspondant aux lignes de la matrice qui lui sont allouées, il a besoin, pour réaliser sa contribution au produit $y = Ax$, des valeurs du vecteur x pour tous les indices de colonnes dans lequel sa matrice locale possède un coefficient non nul. Dans la figure 4.2, ces termes apparaissent gris foncés, les termes gris clairs étant ceux qui ne sont pas nécessaires. Cette méthode de parallélisation permet a priori d'exhiber un parallélisme de degré suffisant correctement équilibré. Il suffit pour cela d'attribuer aux différents processus des blocs de lignes de même taille comprenant approximativement le même nombre de coefficients non nuls. Elle souffre cependant d'un défaut majeur de granularité pour une mise en œuvre sur un système à mémoire distribuée. En effet, le nombre de couples d'opérations $(+, \star)$ pour effectuer le produit par la matrice locale creuse est d'ordre Cn/p, si n est la dimension de la matrice et p le nombre de processeurs, C étant le nombre moyen de coefficients non nuls par ligne. Mais le nombre total de termes du vecteur x à récupérer avant de pouvoir faire le produit peut être d'ordre $(p-1)n/p$, dans le cas où la matrice locale possède des coefficients non nuls dans presque toutes les colonnes, comme dans la figure 4.2. La quantité de données à transférer n'est donc certainement pas petite devant le nombre d'opérations arithmétiques, sauf si l'on trouve le moyen de limiter de manière drastique le nombre de coefficients extérieurs du vecteur x nécessaires pour réaliser le produit par la matrice.

La meilleure approche pour y parvenir consiste à revenir à l'analyse du graphe de la matrice creuse et donc du maillage associé. L'ensemble des sommets associés à une sous-matrice locale détermine un sous-graphe. Les arêtes qui joignent les sommets de ce sous-graphe entre eux représentent des coefficients de la matrice situés

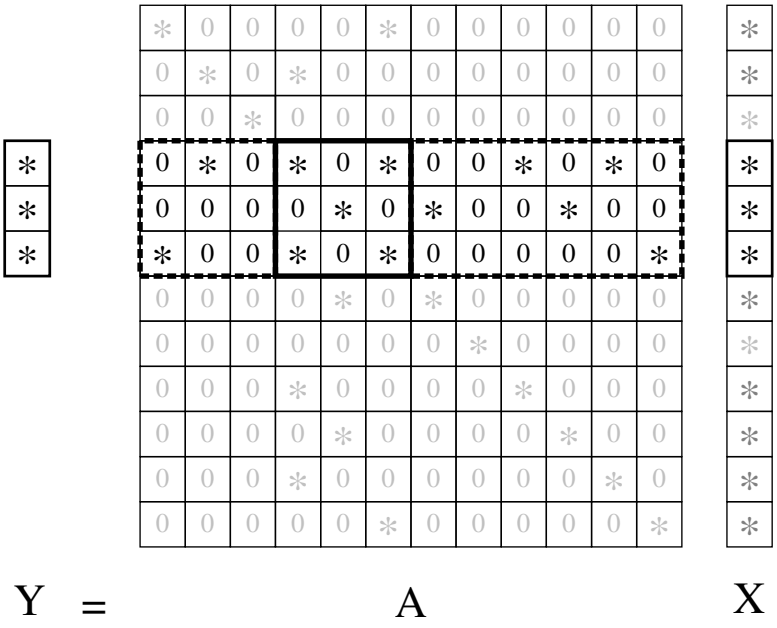

FIGURE 4.2. Produit par un bloc de lignes creux.

dans le bloc diagonal. Lorsque l'on effectue le produit par la matrice, le produit par le bloc diagonal ne demande que des termes locaux du vecteur x. En revanche, les coefficients extra-diagonaux, représentés sur le graphe par des arêtes qui relient des sommets du sous-graphe local à des sommets externes, requièrent les termes du vecteurs x correspondants. Dans la figure 4.3, les arêtes associées à la matrice locale sont représentées en traits pleins, celles qui relient des sommets externes à des sommets internes sont représentées en pointillés.

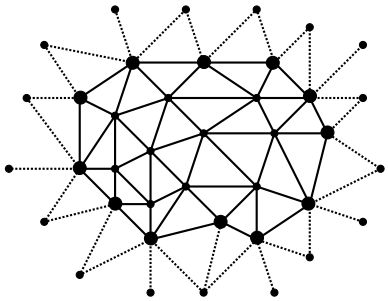

FIGURE 4.3. Sous-graphe associé à un bloc de lignes.

Cette analyse montre clairement que la question de la minimisation des transferts se ramène à un problème de découpage du graphe de la matrice et donc du maillage. Pour que la majorité des arêtes soient internes, ce qui correspond à des coefficients du bloc diagonal, il faut que tous les sommets locaux soient voisins les uns des autres, formant un sous-maillage compact. A l'inverse, les coefficients non locaux du vecteur nécessaires pour réaliser le produit par le bloc de lignes correspondent aux sommets externes voisins des sommets internes, c'est-à-dire à la frontière extérieure de la sous-structure formée par les sommets locaux. Finalement, le découpage optimal est celui qui partitionne le maillage en sous-structures de même taille, pour équilibrer la charge, présentant une frontière la plus réduite possible, pour limiter les transferts de données. Les sous-structures doivent donc être le plus sphérique possible topologiquement, puisque c'est la sphère qui présente la surface extérieure la plus réduite, à volume donné.

14.3 PARALLÉLISATION DU PRODUIT MATRICE-VECTEUR CREUX PAR ENSEMBLE D'ÉLÉMENTS

14.3.1. Rappel des principes du découpage en sous-domaines

Le partitionnement par sommets du maillage présenté dans le paragraphe précédent ne donne pas un véritable découpage en sous-structures géométriques. Les cellules qui contiennent des sommets appartenant à différents sous-ensembles sont nécessaires pour calculer les coefficients d'interaction entre ces sommets dans chacun des processus traitant les blocs de lignes correspondants. Le découpage géométrique du

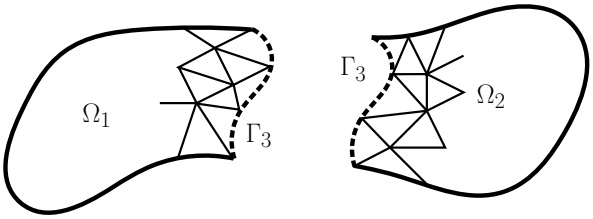

FIGURE 4.4. Maillages distincts des sous-domaines.

domaine spatial correspond à un partitionnement du maillage par cellules. Dans le cas d'un découpage en deux, il apparaît deux sous-domaines, séparés par une interface, comme sur la figure (4.4). La matrice du système global présente alors la structure par blocs suivante :

$$\begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \tag{1}$$

Comme on l'a vu au chapitre sur la factorisation des matrices creuses, l'approche du découpage en sous-domaines présente l'intérêt de conduire à une parallélisation naturelle de la phase de formation de la matrice. En attribuant les différents sous-domaines à des processus distincts, ceux-ci peuvent construire en parallèle les matrices locales :

$$A_1 = \begin{pmatrix} A_{11} & A_{13} \\ A_{31} & A_{33}^{(1)} \end{pmatrix} \qquad A_2 = \begin{pmatrix} A_{22} & A_{23} \\ A_{32} & A_{33}^{(2)} \end{pmatrix} \tag{2}$$

Les blocs $A_{33}^{(1)}$ et $A_{33}^{(2)}$ représentent les interactions entre sommets situés sur l'interface Γ_3 intégrés respectivement dans les sous-domaines Ω_1 et Ω_2, de sorte que :

$$A_{33} = A_{33}^{(1)} + A_{33}^{(2)}.$$

14.3.2. Produit matrice-vecteur

Avec un découpage en deux sous-domaines, le produit matrice-vecteur global s'écrit :

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} A_{11}x_1 + A_{13}x_3 \\ A_{22}x_2 + A_{23}x_3 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 \end{pmatrix}$$

Avec les matrices locales décrite à l'équation (2), on peut calculer indépendamment les deux produits matrice-vecteurs locaux :

$$\begin{pmatrix} y_1 \\ y_3^{(1)} \end{pmatrix} = \begin{pmatrix} A_{11}x_1 + A_{13}x_3 \\ A_{31}x_1 + A_{33}^{(1)}x_3 \end{pmatrix} \qquad \begin{pmatrix} y_2 \\ y_3^{(2)} \end{pmatrix} = \begin{pmatrix} A_{22}x_2 + A_{23}x_3 \\ A_{32}x_2 + A_{33}^{(2)}x_3 \end{pmatrix} \tag{3}$$

Puisque $A_{33} = A_{33}^{(1)} + A_{33}^{(2)}$, $y_3 = y_3^{(1)} + y_3^{(2)}$. Ce qui signifie que le calcul du produit matrice-vecteur complet s'effectue en deux étapes :

(1) produit matrice-vecteur local ;

(2) assemblage sur l'interface des contributions locales.

La première étape ne fait intervenir que des données locales. La seconde nécessite des échanges de données entre processus traitant des sous-domaines possédant une interface commune.

14.3.3. Échanges aux interfaces

Pour assembler les contributions des différents sous-domaines au produit matrice vecteur, chaque processus en charge d'un sous-domaine doit disposer de la description de ses interfaces. Si un sous-domaine Ω_i possède plusieurs sous-domaines voisins, on note Γ_{ij} l'interface entre Ω_i et Ω_j telle qu'elle est décrite dans la processus traitant Ω_i comme dans la figure 4.5. Une interface se décrit simplement par le numéro du sous-domaine voisin et la liste des équations attachées à des points de l'interface. La procédure pour assembler les différentes contributions au calcul du produit matrice-vecteur sur les interfaces se décompose en deux phases.

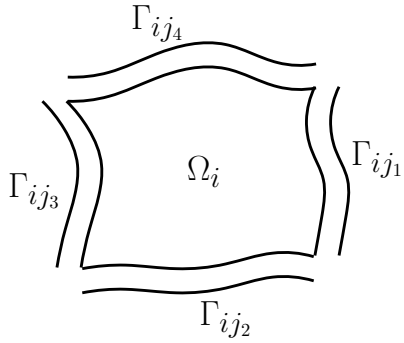

FIGURE 4.5. Description des interfaces.

(1) Pour chaque sous-domaine voisin, ranger les valeurs du vecteur $y = Ax$ local pour toutes les équations interface dans un tableau et l'envoyer au processus en charge du sous-domaine voisin.

```
for s = 1 to nombre_de_voisins
  for i = 1 to n_s
    temp_s(i) = y(list_s(i))
  end for
  Envoyer temp_s à voisin(s)
end for
```

(2) Pour chaque sous-domaine voisin, recevoir le tableau des contributions au produit matrice-vecteur sur l'interface et l'ajouter au composantes du vecteur y correspondant aux équations interface.

```
for s = 1 to nombre_de_voisins
  Recevoir temp_s en provenance de voisin(s)
  for i = 1 to n_s
    y(list_s(i)) = y(list_s(i)) + temp_s(i)
  end for
end for
```

Cette procédure très simple s'applique à n'importe quel nombre de sous-domaines. Si une équation appartient à plusieurs interfaces, le coefficient correspondant du vecteur $y = Ax$ local est envoyé à tous les sous-domaines voisins aux interfaces desquels elle appartient. Inversement, il arrive autant de contributions extérieures qui sont ajoutées au coefficient du vecteur y dans la phase d'assemblage. La procédure pour un nombre quelconque de sous-domaines consiste donc à appliquer pour chaque interface la même procédure que pour deux sous-domaines.
Il n'est pas utile que les équations interface soient numérotées en dernier localement comme dans les équations (2). De même il n'est pas nécessaire de connaître

le numéro des équations interface dans les sous-domaines voisins, puisque seules les valeurs des coefficients d'indices correspondants sont échangés. En revanche, il faut que la liste des équations sur l'interface Γ_{ij} du domains Ω_i soit ordonnée de manière cohérente avec celle de l'interface Γ_{ji} du sous-domaine Ω_j.

14.3.4. Parallélisation du produit scalaire

Contrairement à la méthode de partition par sommets, la méthode de décomposition en sous-domaines oblige à dupliquer les composantes des vecteurs associées aux équations interfaces. Si on calcule les produits scalaires locaux de deux vecteurs x et y dans le cas d'un découpage en deux sous-domaines comme dans la figure 4.4, on obtient les résultats suivants :

$$\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} . \begin{pmatrix} y_1 \\ y_3 \end{pmatrix} = (x_1.y_1) + (x_3.y_3) \qquad \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} . \begin{pmatrix} y_2 \\ y_3 \end{pmatrix} = (x_2.y_2) + (x_3.y_3) \tag{4}$$

La somme des deux contributions locales donne :

$$(x_1 \,.\, y_1) + (x_2 \,.\, y_2) + 2\,(x_3 \,.\, y_3)$$

Dans le cas d'un découpage en plusieurs sous-domaines comme celui de la figure 4.6, ajouter les différentes contributions locales conduit au calcul d'un produit scalaire pondéré pour chaque équation par un facteur égal au nombre de sous-domaines auxquels celle-ci appartient. Ce problème peut-être résolu de différentes façons.

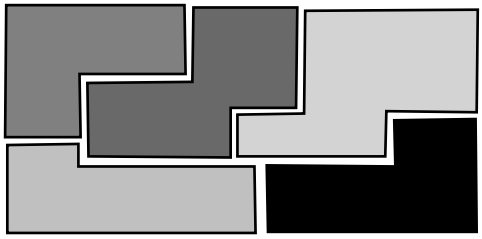

FIGURE 4.6. Découpage multi-domaine.

L'une d'elle consiste à calculer des produits scalaires locaux pondérés pour chaque équation par un facteur égal à l'inverse nombre de sous-domaines auxquels celle-ci appartient. Ce facteur se détermine localement grâce aux listes des équations interfaces. Avec cette technique les produits scalaires locaux de l'équation (4) deviennent :

$$(x_1 \,.\, y_1) + \tfrac{1}{2}\,(x_3 \,.\, y_3) \qquad (x_2 \,.\, y_2) + \tfrac{1}{2}\,(x_3 \,.\, y_3)$$

La sommation des produits scalaires pondérés donne le bon résultat.
Une autre méthode s'appuie sur le fait que, dans les méthodes de Krylov, chaque fois que l'on calcule un produit scalaire, l'un des deux vecteurs au moins est le résultat d'un produit matrice-vecteur. De ce fait, ce vecteur a d'abord été calculé par un

produit matrice-vecteur local, comme le vecteur y de l'équation (3), puis assemblé aux interfaces. L'utilisation des vecteurs non assemblés pour le calcul des produits scalaires locaux donne les contributions suivantes des deux sous-domaines :

$$(x_1 \,.\, y_1) + (x_3 \,.\, y_3^{(1)}) \qquad (x_2 \,.\, y_2) + (x_3 \,.\, y_3^{(2)})$$

La sommation des deux contributions locales conduit au résultat suivant :

$$(x_1 \,.\, y_1) + (x_2 \,.\, y_2) + (x_3 \;.\; y_3^{(1)} + y_3^{(2)}) = (x \,.\, y)$$

Cette méthode s'applique dans le cas d'un découpage multi-domaine quelconque comme dans la figure 4.6. Elle présente l'avantage d'éviter les opérations arithmétiques supplémentaires nécessaires pour la pondération. En revanche elle conduit à des erreurs d'arrondis plus élevées, en particulier lorsque le vecteur assemblé a des coefficients petits, comme c'est le cas du vecteur gradient à convergence.

14.3.5. Application : gradient conjugué parallèle

Avec la méthode de parallélisation par découpage du maillage en sous-domaines, le code du gradient conjugué parallèle dans un environnement de programmation par échange de messages est très semblable au code séquentiel, mis à part qu'il fait appel à deux types d'échanges de message :

(1) échanges de données aux interfaces entre sous-domaines voisins pour l'assemblage des produits matrice-vecteurs ;

(2) opérations de réduction globale pour la sommation des contributions des différents sous-domaines pour les produits scalaires.

Les échanges de données aux interfaces se font à l'aide des fonctions d'échanges point à point de la bibliothèque MPI. Ces échanges sont localisés : chaque sous-domaine n'échange qu'avec ses voisins. Les opérations de réduction impliquent tous les processus et sont réalisées par des appels à la procédure optimisée de la librairie MPI, $MPI\_Allreduce$.

Connaissant la solution approchée à l'itération p, x_p, le vecteur direction de descente, w_p, et le gradient non assemblé aux interfaces, g_p, l'itération $p+1$ du gradient conjugué pour résoudre le système $Ax = b$ requiert les opérations décrites plus bas. La notation des vecteurs en caractères gras signifie qu'il s'agit du produit scalaire global. Le code parallèle ne manipule que les vecteurs locaux.

(1) Produit matrice-vecteur local :

$$v^p = Aw^p$$

(2) Deux produits scalaires locaux utilisant le gradient non assemblé g_p et le produit matrice-vecteur non assemblé v_p :

$$(g^p \,.\, w^p) \;\;,\;\; (v^p \,.\, w^p)$$

(3) Sommation des contributions locales à ces deux produits scalaires afin de calculer le coefficient de descente optimal :

$$\rho^p = -(g^p \,.\, w^p)/(Aw^p \,.\, w^p)$$

(4) Deux combinaisons linéaires pour remettre à jour la solution et le gradient non assemblé :

$$x^{p+1} = x^p + \rho^p w^p$$
$$g^{p+1} = g^p + \rho^p v^p$$

(5) Assemblage aux interfaces pour déterminer le gradient assemblé, g_{ass}^{p+1}.

(6) Produit scalaire local pour calculer le numérateur du coefficient de reconjugaison :

$$(g_{ass}^{p+1} \,.\, v^p)$$

(7) Sommation des contributions locales et calcul du coefficient de reconjugaison :

$$\gamma^p = -(g^{p+1} \,.\, Aw^p)/(Aw^p \,.\, w^p)$$

(8) Combinaison linéaire pour déterminer la nouvelle direction de descente :

$$w^{p+1} = g_{ass}^{p+1} + \gamma^p w^p$$

Le calcul de la norme globale du gradient pour contrôler la convergence se fait en utilisant la même méthodologie.

(1) Produit scalaire local des gradients assemblé et non assemblé, $(g_{ass}^{p+1}.g^{p+1})$.

(2) Sommation des contributions locales pour calculer le carré de la norme $\|\mathbf{g}^{p+1}\|^2$.