Calcul numérique intensif

Jocelyne Erhel - projet ALADIN -

INRIA-RENNES

- Algèbre linéaire dense
- Modules de calcul dans des équations aux dérivées partielles
- Algèbre linéaire creuse
- Résolution de systèmes linéaires creux

Cours IRISA - Février 2001

Algèbre linéaire dense

- opérations de base : BLAS
- exemple : multiplication de matrices
- algorithmes par blocs : LAPACK
- exemple : factorisation de Cholesky
- version parallèle de BLAS : PBLAS
- exemple : produit matrice-vecteur
- version parallèle de LAPACK : SCALAPACK
- bibliographie et logiciels

Opérations de base : BLAS

Basic Algebraic Linear Subroutines (BLAS) : trois niveaux

- BLAS 1 : opérations entre vecteurs
- BLAS 2 : opérations entre matrices et vecteurs
- BLAS 3 : opérations entre matrices

Avantages

- portabilité
- performance : version constructeur
- lisibilité

http://www.netlib.org/blas/index.html

BLAS 1

Opérations entre vecteurs de longueur n
Nombre d'opérations en $O(n)$ et nombre de données en $O(n)$
Exemples : produit scalaire $s = x^T y$, SAXPY $y = y + a * x$

```
NOM
   SAXPY - ajoute le résultat de la multiplication d'un scalaire
           par un vecteur à un autre vecteur (Y := Y + a*X).
SYNOPSIS
   CALL SAXPY(n,sa,sx,incx,sy,incy)

DESCRIPTION
   n    : longueur des vecteurs
   sa   : opérande scalaire
   sx   : vecteur opérande
   incx : incrément sur les composantes de sx
   sy   : vecteur opérande et résultat
   incy : incrément sur les composantes de sy
```

BLAS 2

Opérations matrice-vecteur
Nombre d'opérations en $O(n^2)$ et nombre de données en $O(n^2)$

- produit matrice-vecteur $y = \alpha Ax + \beta y$
- calcul sur matrices triangulaires
- résolution de systèmes triangulaires
- correction de rang 1 (produit extérieur) $A = A + \alpha uv^T$
- correction symétrique de rang 2 $A = A + \alpha(uv^T + vu^T)$

BLAS 3

Opérations matrice-matrice
Nombre d'opérations en $O(n^3)$ et nombre de données en $O(n^2)$
Performance maximale

- produit de matrices $C = A * B + \beta C$
- multiplications de matrices triangulaires
- résolution de systèmes triangulaires à plusieurs seconds membres

Multiplication de matrices

Multiplication : $C = C + A * B$

- `A matrice n1 x n2`
- `B matrice n2 x n3`
- `C matrice n1 x n3`

```
for i = 1:n1,
for j = 1:n3,
for k = 1:n2,
    C(i,j) = C(i,j) + A(i,k)*B(k,j);
end;
```

$n1 * n2 * n3$ opérations

6 multiplications de matrices

Trois boucles commutables $\Rightarrow$ 6 écritures

1 boucle interne : BLAS 1

(i,j,k) ou (j,i,k) : produit scalaire ligne-colonne
(i,k,j) ou (k,i,j): saxpy sur des lignes
(j,k,i) ou (k,j,i) : saxpy sur des colonnes

2 boucles internes : BLAS 2

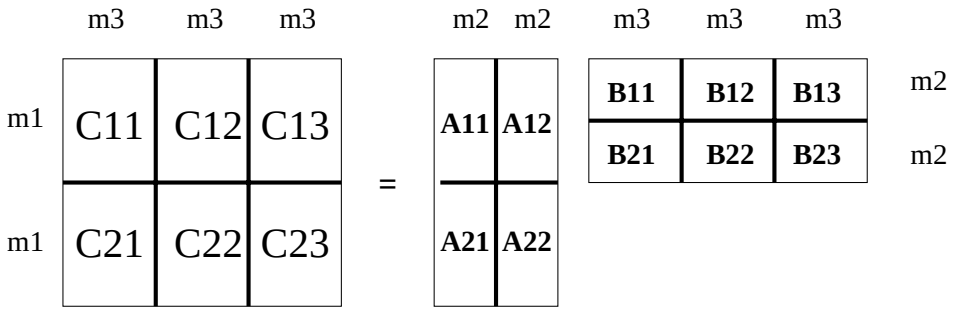
(i,j,k) ou (i,k,j) : produit ligne-matrice
(j,i,k) ou (j,k,i): produit matrice-colonne
(k,i,j) ou (k,j,i) : correction de rang 1

Analyse des lectures

cas $n_1 = n_2 = n_3$ - version (i,j,k)
si B dans le cache, $O(n^2)$ lectures
sinon $O(n^3)$ lectures

Version par blocs pour utiliser le cache

Multiplication par blocs



n1 = k1 x m1, n2 = k2 x m2, n3 = k3 x m3
version (ikj)

```
for i = 1:k1
for k = 1:k2,
for j = 1:k3,
    Cij = Cij + Aik x Bkj;
end;
```

Analyse des lectures dans la version par blocs

un bloc A_{ik} dans la mémoire cache
A_{ik} lu une fois - A lu une fois - $n_1 n_2$ mots
B_{kj} lu k_1 fois - B lu k_1 fois - $k_1 n_2 n_3$ mots
C_{ij} lu k_2 fois - C lu k_2 fois - $k_2 n_1 n_3$ mots

Total : $L = n_1 n_2 + k_1 n_2 n_3 + k_2 n_1 n_3 = n_1 n_2 + n_1 n_2 n_3 \left(\frac{1}{m_1} + \frac{1}{m_2}\right)$.

Taille optimale des blocs

Mémoire cache de $M + 1 bloc$ mots : $m_1 m_2 \leq M$

Stratégie optimale : $m_1 m_2 = min(M, n_1 n_2)$

Si $n_1 n_2 \leq M$**:** alors $m_1 = n_1$ et $m_2 = n_2$
sinon si $n_1 \leq \sqrt{M}$ **et** $n_2 > \sqrt{M}$**:** alors $m_1 = n_1$ et $m_2 = \frac{M}{m_1}$
sinon si $n_2 \leq \sqrt{M}$ **et** $n_1 > \sqrt{M}$**:** alors $m_2 = n_2$ et $m_1 = \frac{M}{m_2}$
sinon : $m_1 = m_2 = \sqrt{M}$

version BLAS optimisée pour chaque architecture

LAPACK

Résolution de problèmes d'algèbre linéaire

- Systèmes linéaires
- Problèmes linéaires aux moindres carrés
- Problèmes aux valeurs propres
- Problèmes aux valeurs singulières
- Estimation de conditionnement

Sur des matrices pleines ou bandes

Versions par blocs

BLAS3 le plus souvent possible
sinon BLAS2 et BLAS1

http://www.netlib.org/lapack/

Factorisation de Cholesky

$A \in \mathbb{R}^{n \times n}$ matrice symétrique $A^T = A$ définie positive $\forall x \neq 0,\ x^T A x > 0$

Factorisation de Cholesky $A = LL^T$ et $diag(L) > 0$

version récursive

$$A = \begin{pmatrix} \alpha & a^T \\ a & A_1 \end{pmatrix} \qquad L = \begin{pmatrix} \lambda & 0 \\ l & L_1 \end{pmatrix}$$

$$\begin{cases} \lambda = \sqrt{\alpha} \\ l = (1/\lambda)a \\ L_1 L_1^T = A_1 - l\ l^T \end{cases}$$

3 niveaux : 6 manières de dérouler la récursion

Cholesky : Right Looking ou Fan-out ou jki

$L :=$ triangle_inférieur(A);
for $j = 1,n,$
    $L(j,j) = \sqrt{L(j,j)}$;
    for $i = j+1,n$
        $L(i,j) := L(i,j)/L(j,j)$;
    end;
    for $k = j+1,n$
        for $i = k,n$
            $L(i,k) := L(i,k) - L(i,j) * L(k,j)$;
        end;
    end;
end

BLAS2 et BLAS1

Cholesky : Left Looking ou Fan-in ou kji

$L :=$ triangle_inférieur(A);
for $k = 1,n,$
    for $j = 1,k-1$
        for $i = k,n$
            $L(i,k) := L(i,k) - L(i,j) * L(k,j)$;
        end;
    end;
    $L(k,k) := \sqrt{L(k,k)}$;
    for $i = k+1,n$
    $L(i,k) = L(i,k)/L(k,k)$;
end

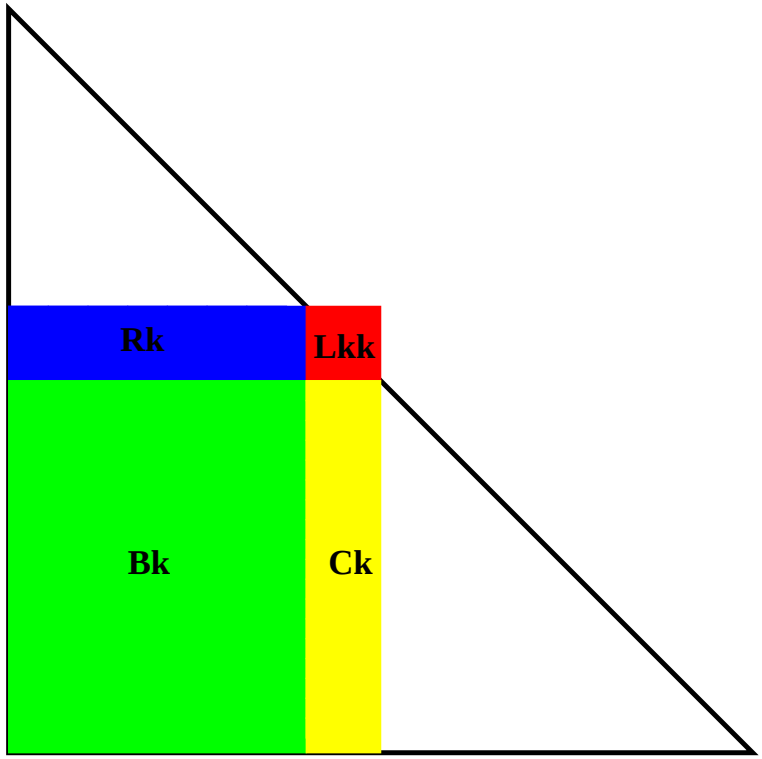
BLAS2 et BLAS1

Cholesky par blocs - Fan-in

$L =$ triangle_inférieur(A);
for $k = 1,nblocs,$
    for $j = 1,k-1$
        for $i = k,nblocs$
            $L(i,k) = L(i,k) - L(i,j) * L(k,j)^T$;
        end;
    end;
    $L(k,k) = chol(L(k,k))$;
    for $i = k+1,nblocs$
    $L(i,k) = L(i,k) * L(k,k)^{-T}$;
end

BLAS3 et BLAS2

Cholesky par blocs - Fan-in



$L =$ triangle_inférieur(A);
for $k = 1, nblocs,$
$L_{kk} = L_{kk} - R_k * R_k^T$;
$L_{kk} = chol(L_{kk})$;
$C_k = C_k - B_k * R_k^T$;
$C_k = C_k * L_{kk}^{-T}$;
end

Parallel BLAS

Mémoire partagée

version parallèle de BLAS

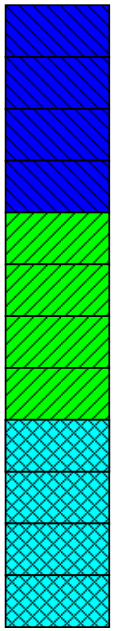
Mémoire distribuée

PBLAS : Version distribuée de BLAS avec distribution des vecteurs et des matrices communications par la bibliothèque BLACS

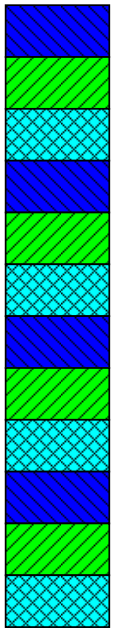
Distribution de vecteurs

P processus répartis sur une colonne - Vecteur de longueur N

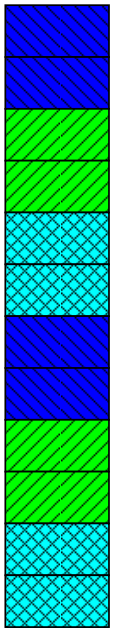
par blocs



cyclique



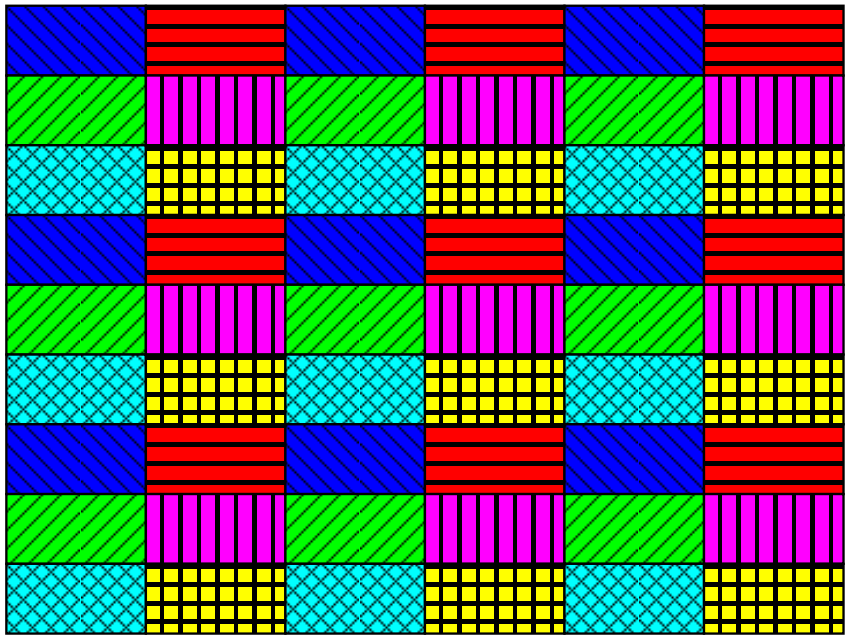
bloc-cyclique



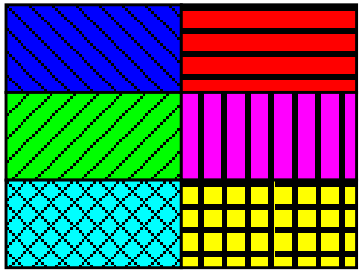
paramètre d'optimisation : taille des blocs r

Distribution de matrices

distribution bloc-cyclique - Matrice de taille $M \times N$



grille de $P = P_r \times P_c$ processus



paramètres d'optimisation :

tailles des blocs $r \times c$ et de la grille $P_r \times P_c$

produit matrice-vecteur $y = A \times x$

partition par blocs de taille $r \times c$ avec $M = rM_r$,$N = cN_c$

$Y_i = \sum_{j=1}^{N_c} A_{ij}\ X_j,\ i = 1,M_r$

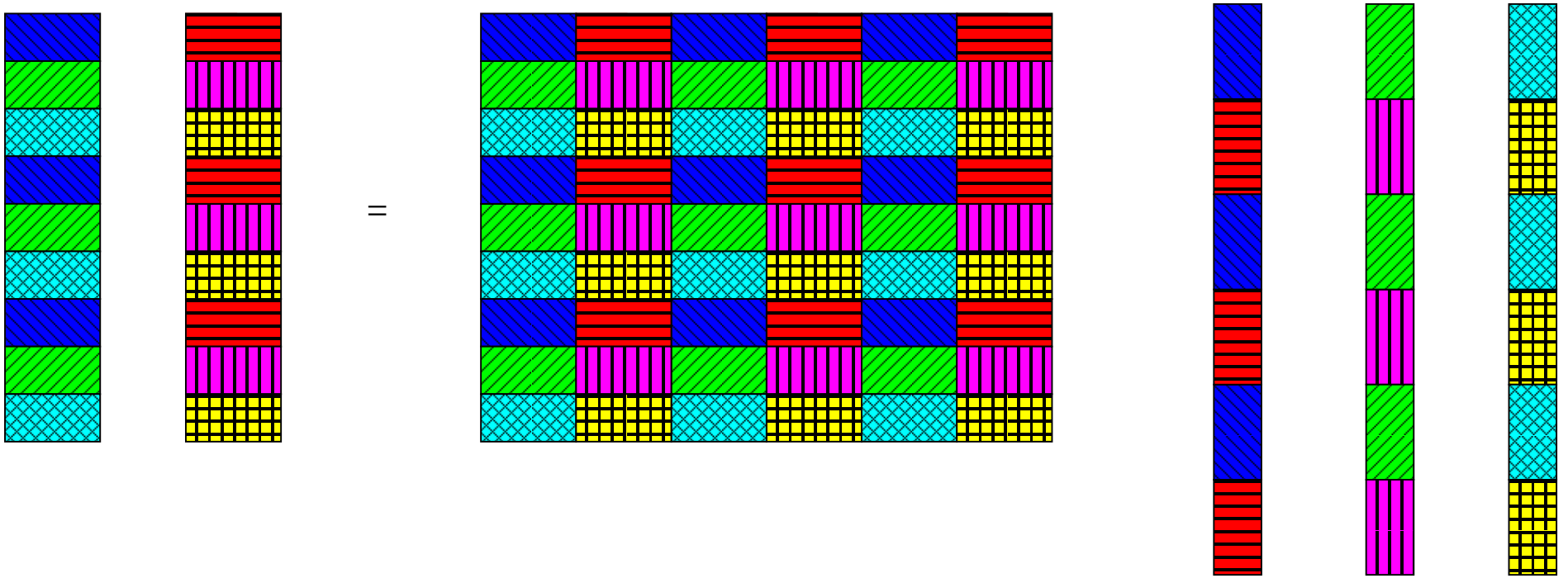
Grille de $P = P_r \times P_c$ processus

Distribution bloc-cyclique de A

Distribution bloc-cyclique de X sur chacune des lignes de la grille

Distribution bloc-cyclique de Y sur chacune des colonnes de la grille

Version distribuée



Algorithme de chaque processus

Processus (p,q)
si $p > 1$ alors
recevoir $X_j, \quad j = q \ mod \ P_c$
sinon
diffuser $X_j, \quad j = q \ mod \ P_c$ à $\{(p,q), \quad p > 1\}$
finsi
calculer $Y_{iq} = \sum_{j=q \ mod \ P_c} A_{ij} \ X_j, \ i = p \ mod \ P_r$
si $q > 1$ alors
envoyer Y_{iq} à $(p,1)$
sinon
recevoir $\{Y_{iq}, q > 1\}$ de $\{(p,q), \quad q > 1\}$
$Y_i = \sum_{q=1}^{P_c} Y_{iq}$
finsi

Volume de calcul et de communications

Calcul : $O(MN/P)$: équilibré

Communications : $O(NP_r/P_c + MP_c/P_r)$

$M = N$: minimiser $P_r/P_c + P_c/P_r$ avec $P_cP_r = P$

$M = N$: communications minimales si $P_r = P_c = \sqrt{P}$

Bibliographie et logiciels

Bibliographie

- Lapack Working Notes et Lapack user's guide
- G. Golub and C. van Loan, Matrix Computations, John Hopkins University Press, 3rd edition, 1996

Logiciels

- BLAS : http://www.netlib.org/blas/index.html
- LAPACK : http://www.netlib.org/lapack/
- MATLAB :
- SCILAB :
- BLACS : http://www.netlib.org/blacs/index.html
- PBLAS : http://www.netlib.org/scalapack/html/pblas_qref.html
- SCALAPACK : http://www.netlib.org/scalapack/scalapack_home.html

Modules de calcul
dans des équations aux dérivées partielles

- Un exemple : l'équation de la chaleur
- Discrétisation des EDP
- Résolution de systèmes non linéaires
- Evaluation de fonctions non linéaires

Équation de la chaleur - discrétisation en espace

Modèle continu

$$\begin{cases} \frac{\partial u}{\partial t} + a\frac{\partial^2 u}{\partial x^2} + b\frac{\partial^2 u}{\partial y^2} = f \quad \text{dans } \Omega \times [0,T], \\ u = g \quad \text{sur } \Gamma, \\ u(0,x,y) = u_0(x,y) \quad \text{dans } \Omega. \end{cases}$$

Discrétisation en espace

Différences finies sur une grille régulière de pas h

$$\frac{\partial^2 u}{\partial x^2}(x_i,y_i) \simeq \frac{u(x_{i+1},y_i) + u(x_{i-1},y_i) - 2u(x_i,y_i)}{h^2}$$

Équation de la chaleur - équation discrétisée

Système d'équations différentielles ordinaires (EDO)

$$\begin{cases} \frac{dU}{dt} - AU = F, \\ U(0) = U_0. \end{cases}$$

A matrice creuse à structure régulière liée au maillage

Équation de la chaleur - discrétisation en temps

Schéma en temps explicite - Par exemple, Euler

$$\begin{cases} \frac{U_{n+1}-U_n}{dt} - AU_n = F_n, \\ U_{n+1} = (I + dtA)U_n + dtF_n \end{cases}$$

Produit matrice creuse par vecteur

Schéma en temps implicite - Par exemple, Euler

$$\begin{cases} \frac{U_{n+1}-U_n}{dt} - AU_{n+1} = F_n, \\ (I - dtA)U_{n+1} = U_n + dtF_n \end{cases}$$

Résolution de système linéaire creux

Equations aux dérivées partielles (EDP)

Modèle continu

$$\begin{cases} \frac{\partial u}{\partial t} + \phi(u) = f \ \text{ dans } \Omega \times [0,T], \\ \text{conditions aux limites sur } \Gamma, \\ u(0) = u_0 \ \text{ dans } \Omega. \end{cases}$$

Discrétisation en espace

Eléments finis, volumes finis, etc sur un maillage quelconque

Système d'équations différentielles ordinaires (EDO)

$$\begin{cases} \frac{dU}{dt} + \Phi(U) = F, \\ U(0) = U_0. \end{cases}$$

Φ fonction non linéaire à structure liée au maillage

EDP - discrétisation en temps

Schéma en temps explicite - Par exemple, Euler

$$\begin{cases} \frac{U_{n+1}-U_n}{dt} + \Phi(U_n) = F_n, \\ U_{n+1} = U_n + dt\Phi(U_n) + dtF_n \end{cases}$$

Evaluation d'une fonction non linéaire

Schéma en temps implicite - Par exemple, Euler

$$\begin{cases} \frac{U_{n+1}-U_n}{dt} + \Phi(U_{n+1}) = F_n, \\ U_{n+1} + dt\Phi(U_{n+1}) = U_n + dtF_n \end{cases}$$

Résolution de système non linéaire

Résolution de système non linéaire

$\Phi(U) = 0$

Méthode de Newton

$J(U_k)(U_{k+1} - U_k) = -\Phi(U_k)$

$J(U_k)$ Jacobien du système

Evaluation de fonction non linéaire

Evaluation de matrice creuse

Résolution de système linéaire creux

Evaluation d'une fonction non linéaire

calcul explicite

pas de dépendances en lecture-écriture

seulement des dépendances en lecture : graphe et matrice associée

graphe non connecté

exemple : calcul vectoriel

distribution comme en algèbre linéaire dense

graphe d'un maillage

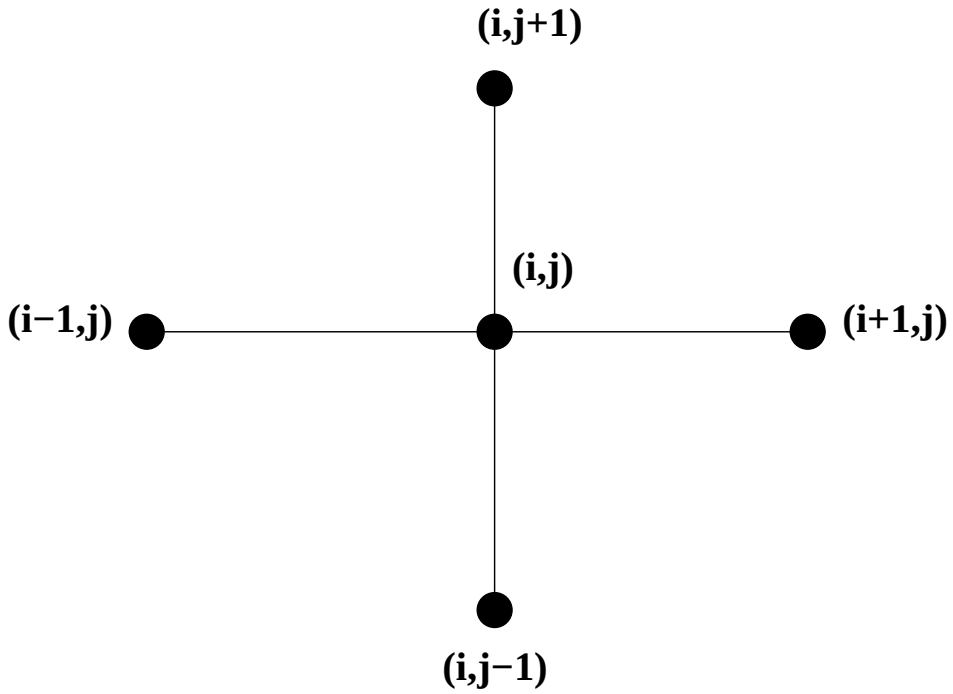
exemple : schéma en temps explicite et discrétisation en espace

distribution comme en algèbre linéaire creuse

Algèbre linéaire creuse

- produit matrice creuse par vecteur - cas structuré
- stockage de matrices creuses non structurées
- produit matrice creuse par vecteur - cas non structuré
- bibliographie et logiciels

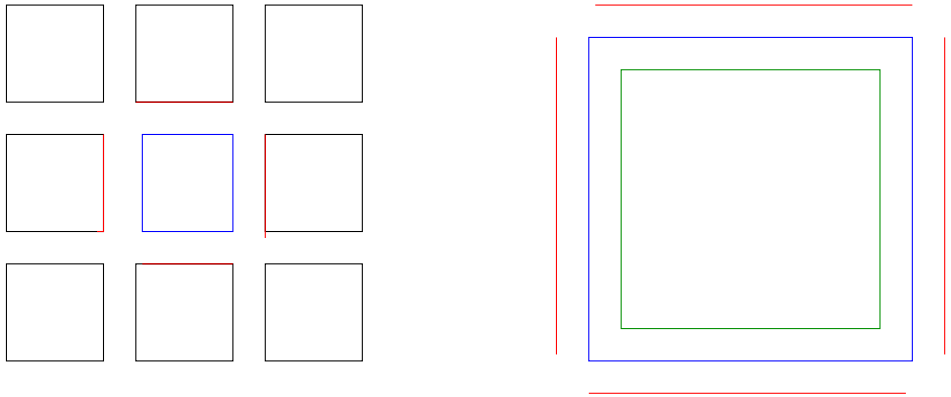
Produit matrice creuse par vecteur - cas structuré



Différences finies à 5 points

$$v_{ij} = 4u_{ij} - u_{i-1j} - u_{i+ij} - u_{ij-1} - u_{ij+1}$$

Distribution par sous-domaines



Un sous-domaine

Points internes

Points frontières

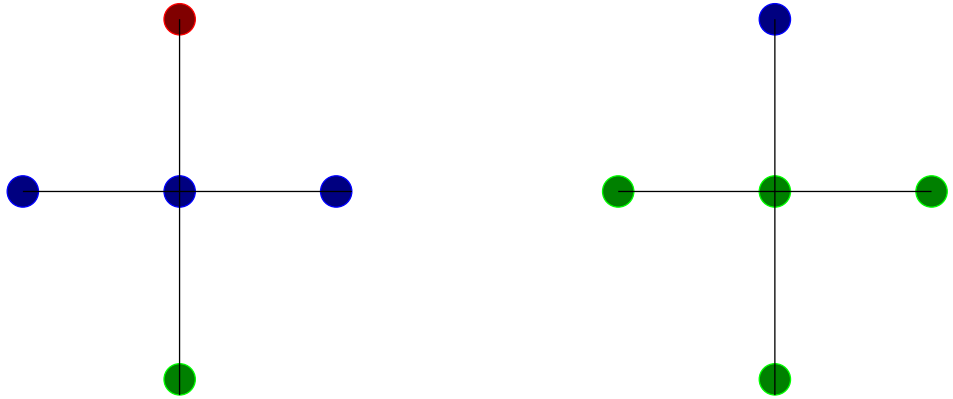
Les sous-domaines voisins

Points externes voisins

Parallélisme à l'intérieur des sous-domaines

Communications aux frontières des sous-domaines

Calcul sur chaque sous-domaine



point frontière point interne

Algorithme pour chaque sous-domaine

Pour chaque voisin, Faire
    Recevoir les points externes voisins
    Calculer les points frontières
    Envoyer les points frontières
Fait
Calculer les points internes

Recouvrement des calculs par les communications

Volumes de calcul et de communications

Volume de calcul = O(nombre de points) = O(volume)
Volume de communications = O(nombre de points frontières) = O(surface)

Équilibrer les volumes des sous-domaines
Minimiser les surfaces des frontières

Partition et allocation

Allocation homogène de p sous-domaines sur p processeurs

Rectangle de $m \times n$ points et p sous-domaines
Partition simple en sous-rectangles $m/k_1 \times n/k_2$
Volume d'un sous-rectangle $= mn/k_1k_2$
Surface d'un sous-rectangle $= m/k_1 + n/k_2$

Optimisation

Équilibrer les volumes : $mn/k_1k_2 = mn/p \Leftrightarrow k_1k_2 = p$
Minimiser les surface : $min_{k_1}(m/k_1 + nk_1/p)$
Solution : $k_1 = \sqrt{mp/n}$, $k_2 = \sqrt{np/m}$

Domaine carré de côté n : sous-domaine carré de côté $n/\sqrt{p}$

Code de calcul d'un processeur

Données

taille du domaine complet M,N

position du coin inférieur gauche (ic,jc)

points : $(1:I,1:J)$

points internes : $(2:I-1,2:J-1)$

points frontière : $(1:I,1),(1:I,J),(1,1:J),(I,1:J)$

points externes : $sud(1:I),nord(1:I),ouest(1:J),est(1:J)$

Code exécutable

Si $ic > 1$ alors
  Recevoir $ouest(1:J)$
  Calculer $u(1,1:J)$
  Envoyer $u(1:J)$
Finsi
...Même chose pour les 3 autres frontières
Calculer les points internes $u(2:I-1,2:J-1)$

Produit matrice creuse par vecteur - cas non structuré

différents stockages
produit matvec CSR
produit matvec parallèle

Résolution de système linéaire creux

- Méthodes directes
- Méthodes itératives classiques
- Méthodes itératives de Krylov
- Méthodes multigrilles
- Méthodes de sous-domaines (semi-directes)

Méthodes directes

Factorisation de Cholesky

remplissage

renumérotation

factorisation symbolique

Factorisation de Gauss

critère de stabilité

Versions parallèles

multifrontale

Méthodes itératives classiques

décomposition $A = M - N$

Jacobi

convergence très très lente

Gauss-Seidel

convergence très lente

SSOR

cas symétrique défini positif

Versions par blocs

souvent plus performant

Méthodes itératives de Krylov

- Cas symétrique défini positif
- Cas symétrique indéfini
- Cas non symétrique
- Préconditionnements

A symétrique définie positive

Une méthode de choix : Gradient Conjugué (CG)

- algorithme
- propriétés
- convergence
- préconditionnement

Gradient Conjugué

Algorithme

Initialisation

choix de x_0

$p_0 = r_0 = b - Ax_0$

Pour $k = 0,1 \ldots$

$\alpha_k = \frac{\|r_k\|^2}{(Ap_k,p_k)}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$\beta_{k+1} = \frac{\|r_{k+1}\|^2}{\|r_k\|^2}$

$p_{k+1} = r_{k+1} + \beta_{k+1} p_k$

Fin Pour

Propriétés

$(r_{k+1},p_k) = 0$

$(r_{k+1},r_k) = 0$

$(p_{k+1},Ap_k) = 0$

$\|r_{k+1}\|_{A^{-1}} = \min_\alpha \|r_k - \alpha A p_{k-1}\|_{A^{-1}}$

Gradient Conjugué - propriétés

Propriétés d'orthogonalité et de minimisation

$$(r_k,p_i) = (r_k,r_i) = 0, \;\; i \leq k-1$$
$$(p_k,Ap_i) = 0, \;\; i \leq k-1$$
$$\|r_{k+1}\|_{A^{-1}} \leq \|r_k\|_{A^{-1}}$$
$$\|r_k\|_{A^{-1}} = \min_{x \in x_0 + Span(p_0,\ldots,p_{k-1})} \|b - Ax\|_{A^{-1}}$$

Méthode de Krylov

$$\mathcal{K}_k(A,r_0) = Span(r_0,Ar_0,\ldots,A^{k-1}r_0) \quad \text{Espace de Krylov}$$
$$\mathcal{K}_k(A,r_0) = Span(r_0,r_1,\ldots,r_{k-1}) = Span(p_0,p_1,\ldots,p_{k-1})$$

Méthode de projection

$$x_k \in x_0 + \mathcal{K}_k(A,r_0) \quad \text{Condition d'espace}$$
$$r_k \perp \mathcal{K}_k(A,r_0) \quad \text{Condition de Galerkin}$$

Gradient Conjugué - convergence

Méthode polynomiale

$x_k = x_0 + Q_{k-1}(A) r_0$
$r_k = (I - A Q_{k-1}(A)) r_0 = P_k(A) r_0$ avec $P_k(0) = 1$

Propriété minmax - Convergence asymptotique

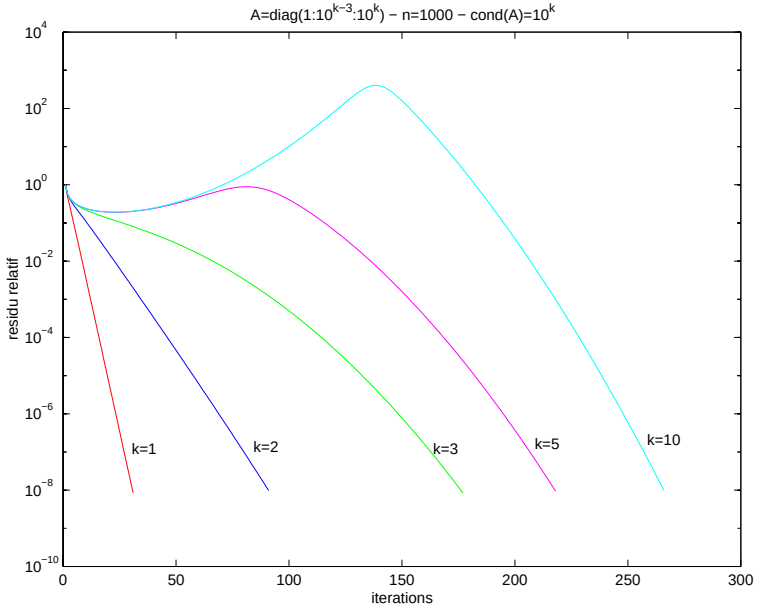
$A = V \Delta V^{-1}$ avec $\Delta = diag(\lambda_1, \ldots, \lambda_n)$
$0 < \lambda_1 \leq \ldots \leq \lambda_n$ et $\kappa(A) = \lambda_n / \lambda_1$

$$\|r_k\|_{A^{-1}} \leq \|r_0\|_{A^{-1}} \max_{\{\lambda_j\}} |P_k(\lambda_j)|$$
$$\leq \|r_0\|_{A^{-1}} \min_{\{P/deg(P)=k \; P(0)=1\}} \max_{\lambda_1 \leq t \leq \lambda_n} |P(t)|$$

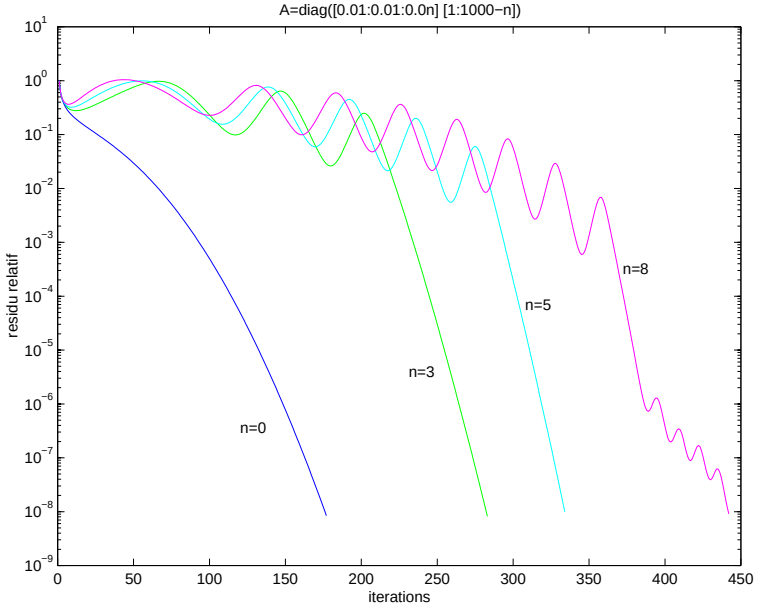
$$\|r_k\|_{A^{-1}} \leq 2 \|r_0\|_{A^{-1}} \left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1} \right)^k$$

Gradient Conjugué - exemples

Convergence asymptotique



Convergence superlinéaire



Gradient Conjugué Préconditionné

Préconditionnement M symétrique défini positif

Algorithme

Initialisation

choix de x_0

$r_0 = b - Ax_0$

$z_0 = M^{-1} r_0$

$p_0 = z_0$

Pour $k = 0,1 \ldots$

$\alpha_k = \frac{(r_k, z_k)}{(Ap_k, p_k)}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$z_{k+1} = M^{-1} r_{k+1}$

$\beta_{k+1} = \frac{(r_{k+1}, z_{k+1})}{(r_k, z_k)}$

$p_{k+1} = z_{k+1} + \beta_{k+1} p_k$

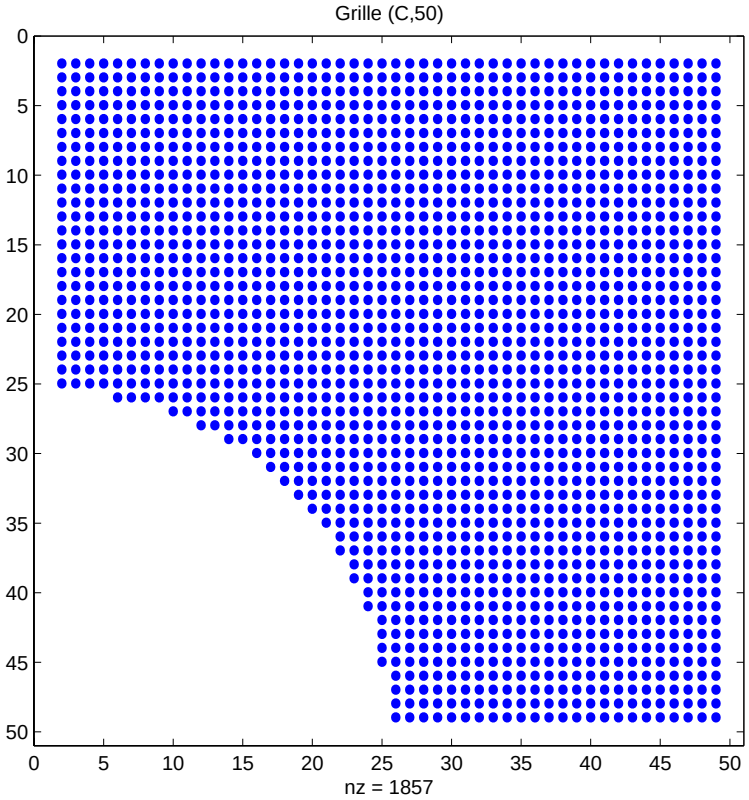
Fin Pour

Exemples

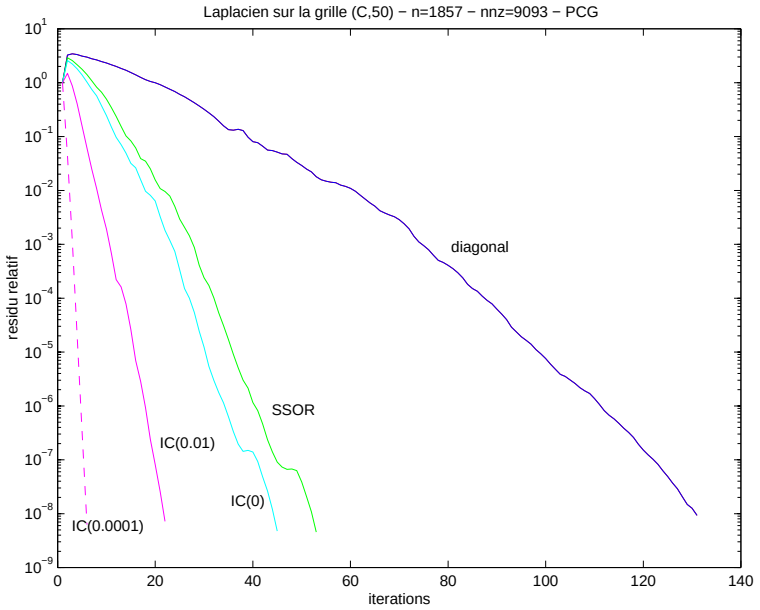
- diagonal
- SSOR
- polynomial
- IC(k)
- IC(drop)
- inverse approché
- m-step SSOR
- multigrille
- multi-niveau
- etc

Gradient Conjugué Préconditionné- exemples

Grille de différences finies



Courbes de convergence



Gradient Conjugué Préconditionné - dépendances

Algorithme

Pour $k = 0,1 \ldots$

$q_k = Ap_k$

$\alpha_k = \frac{(r_k,z_k)}{(q_k,p_k)}$

$x_{k+1} = x_k + \alpha_k p_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$z_{k+1} = M^{-1} r_{k+1}$

$\beta_{k+1} = \frac{(r_{k+1},z_{k+1})}{(r_k,z_k)}$

$p_{k+1} = z_{k+1} + \beta_{k+1} p_k$

Fin Pour

Opérations

produit matrice vecteur

produit scalaire

opération vectorielle

opération vectorielle

système linéaire

produit scalaire

opération vectorielle

Enchaînement séquentiel d'opérations vectorielles et matricielles

Lanczos symétrique

Procédé de Lanczos symétrique

$Span(V_k) = \mathcal{K}_k(A, v_1)$ Base de l'espace de Krylov

$V_k^T V_k = I$ Système orthonormé

$$AV_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T \qquad T_k = \begin{pmatrix} \gamma_1 & \delta_2 & & \\ \delta_2 & \gamma_2 & \ddots & \\ & \ddots & \ddots & \delta_k \\ & & \delta_k & \gamma_k \end{pmatrix}$$

Lien avec Gradient Conjugué

$r_0 = \|r_0\|_2 v_1 = \beta v_1$ Espace de Krylov

$x_k = x_0 + V_k y \quad y \in \mathbb{R}^k$ Condition d'espace

$r_k = r_0 - AV_k y = V_k(\beta e_1 - T_k y) - \delta_{k+1}(e_k^T y) v_{k+1}$

$V_k^T r_k = 0 \Leftrightarrow T_k y = \beta e_1$ Condition de Galerkin

Gradient Conjugué : Factorisation de Cholesky de T_k
et propriété de minimisation de $\|r_k\|_{A^{-1}}$

A symétrique indéfinie

SYMMLQ, MINRES

Lanczos symétrique avec moindres carrés

A non symétrique

Cela se complique ...

Pas possible d'avoir à la fois
une récurrence courte et
une minimisation de la norme du résidu

- Récurrence courte : Gradient Bi-Conjugué, QMR, etc
- Minimisation : GMRES
- Préconditionnement

Gradient Bi-Conjugué

Algorithme

Initialisation

choix de x_0 et $\tilde{x}_0$

$r_0 = b - Ax_0$ et $\tilde{r}_0 = b - A^T\tilde{x}_0$

$p_0 = r_0$ et $\tilde{p}_0 = \tilde{r}_0$

Pour $k = 0,1\ldots$

$\alpha_k = \frac{(r_k,\tilde{r}_k)}{(Ap_k,\tilde{p}_k)}$

$x_{k+1} = x_k + \alpha_k p_k$

$\tilde{x}_{k+1} = \tilde{x}_k + \alpha_k \tilde{p}_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$\tilde{r}_{k+1} = \tilde{r}_k - \alpha_k A^T \tilde{p}_k$

$\beta_{k+1} = \frac{(r_{k+1},\tilde{r}_{k+1})}{(r_k,\tilde{r}_k)}$

$p_{k+1} = r_{k+1} + \beta_{k+1} p_k$

$\tilde{p}_{k+1} = \tilde{r}_{k+1} + \beta_{k+1}\tilde{p}_k$

Fin Pour

Gradient Bi-Conjugué - propriétés

Propriétés d'orthogonalité

$(r_k,\tilde{r}_i) = 0,\ \ i \leq k-1$
$(\tilde{p}_k,Ap_i) = 0,\ \ i \leq k-1$

pas de propriété de minimisation

Méthode de Krylov

$\mathcal{K}_k(A,r_0) = Span(r_0,Ar_0,\ldots,A^{k-1}r_0)$ Espace de Krylov associé à A
$\mathcal{K}_k(A^T,\tilde{r}_0) = Span(\tilde{r}_0,A^Tr_0,\ldots,(A^T)^{k-1}\tilde{r}_0)$ Espace de Krylov associé à A^T

Méthode de projection

$x_k \in x_0 + \mathcal{K}_k(A,r_0)$ Condition d'espace
$r_k \perp \mathcal{K}_k(A^T,\tilde{r}_0)$ Condition de Galerkin

Lanczos non symétrique

Procédé de Lanczos non symétrique

$Span(V_k) = \mathcal{K}_k(A, v_1)$ Base de l'espace de Krylov

$Span(W_k) = \mathcal{K}_k(A^T, w_1)$ Base de l'espace de Krylov dual

$W_k^T V_k = I$ Système bi-orthogonal

$$AV_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T$$

$$T_k = \begin{pmatrix} \gamma_1 & \eta_2 & & \\ \delta_2 & \gamma_2 & \ddots & \\ & \ddots & \ddots & \eta_k \\ & & \delta_k & \gamma_k \end{pmatrix}$$

$$A^T W_k = W_k T_k^T + \eta_{k+1} w_{k+1} e_k^T$$

Lien avec Gradient Bi-Conjugué

$r_0 = \|r_0\|_2 v_1 = \beta v_1$ Espace de Krylov

$x_k = x_0 + V_k y \quad y \in \mathbb{R}^k$ Condition d'espace

$r_k = r_0 - AV_k y = V_k(\beta e_1 - T_k y) - \delta_{k+1}(e_k^T y) v_{k+1}$

$W_k^T r_k = 0 \Leftrightarrow T_k y = \beta e_1$ Condition de Galerkin

Gradient Bi-Conjugué : Factorisation de Gauss de T_k

BCG - Convergence - Variantes

risque de Breakdown dans Lanczos : $(r_k, \tilde{r}_k) = 0$
Version de Lanczos avec Look-Ahead

Convergence irrégulière
Produit par A^T : version Transpose-Free BICGSTAB
Convergence plus régulière de BICGSTAB

risque de Breakdown dans LU : $(\tilde{p}_k, Ap_k) = 0$
Algorithme QMR

Résidu Quasi-minimum QMR

Algorithme

Lanczos non symétrique avec look-ahead

$r_k = V_{k+1}(\beta e_1 - \overline{T}_k y)$ $\qquad$ $\overline{T}_k = \begin{pmatrix} T_k \\ \delta_{k+1} e_k^T \end{pmatrix}$

$\Omega_{k+1} = diag(\|v_1\|_2, \ldots, \|v_{k+1}\|_2)$

Résoudre $\min_{y \in \mathbb{R}^k} \|\beta e_1 - \Omega_{k+1} \overline{T}_k y\|_2$

Convergence

pas de breakdown

SI T_k est diagonalisable

$\|r_k\|_2 \leq \|r_0\|_2 \sqrt{k+1}\, \kappa(T_k) \min_{\{P/deg(P)=k\ P(0)=1\}} \max_{1 \leq j \leq n} |P(\lambda_j)|$

QMR préconditionné - algorithme

version sans look-ahead

QMR préconditionné - dépendances

Enchaînement séquentiel d'opérations vectorielles et matricielles

Arnoldi et GMRES

Procédé d'Arnoldi

$Span(V_k) = \mathcal{K}_k(A, v_1)$ Base de l'espace de Krylov

$V_k^T V_k = I$ Système orthonormé

$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T$ H_k matrice de Hessenberg

Algorithme de GMRES

$r_0 = \|r_0\|_2 v_1 = \beta v_1$ Espace de Krylov

$x_k = x_0 + V_k y \;\; y \in \mathbb{R}^k$ Condition d'espace

$r_k \perp AV_k$ Condition de Galerkin

$$r_k = r_0 - AV_k y = V_{k+1}(\beta e_1 - \overline{H}_k y) \qquad \overline{H}_k = \begin{pmatrix} H_k \\ h_{k+1,k} e_k^T \end{pmatrix}$$

$$(AV_k)^T r_k = \overline{H}_k^T (\beta e_1 - \overline{H}_k y)$$

Résoudre $\min_{y \in \mathbb{R}^k} \|\beta e_1 - \overline{H}_k y\|_2$ Condition de Galerkin

GMRES - Convergence

Méthode polynomiale

$$x_k = x_0 + Q_{k-1}(A)r_0$$
$$r_k = (I - AQ_{k-1}(A))r_0 = P_k(A)r_0 \text{ avec } P_k(0) = 1$$

Propriété minmax

SI $A = U\Delta U^{-1}$ avec $\Delta = diag(\lambda_1, \ldots, \lambda_n)$

$$\|r_k\|_2 \leq \|r_0\|_2 \kappa(U) \min_{\{P/deg(P)=k\ P(0)=1\}} \max_{1 \leq j \leq n} |P(\lambda_j)|$$

GMRES - redémarrage

Complexité et stockage

k itérations

Arnoldi : $O(k \times nz) + O(k^2 \times n)$ opérations

Moindres carrés négligeable

Stockage de $k + 3$ vecteurs de longueur n

Redémarrage : GMRES(k)

Initialisation

choix de x_0

Jusqu'à convergence

k itérations de GMRES

$x_k = x_0 + V_k y$

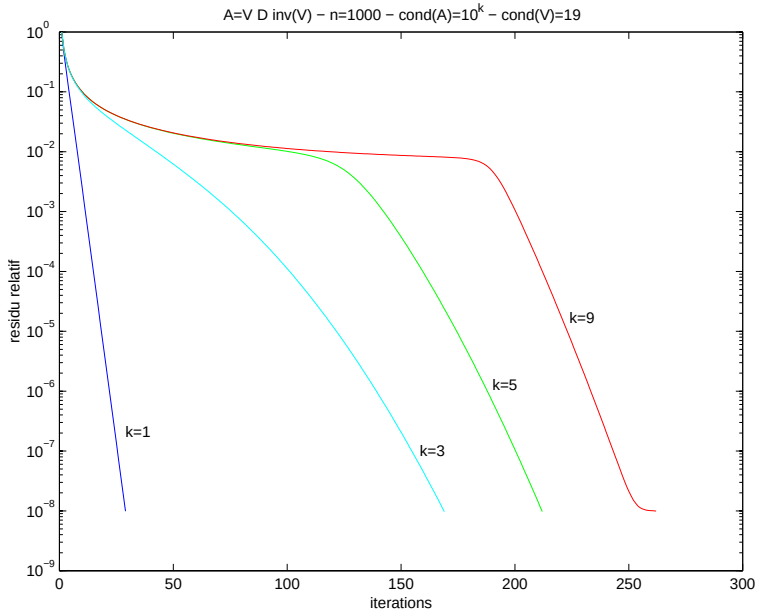
$x_0 = x_k$

Fin Jusque

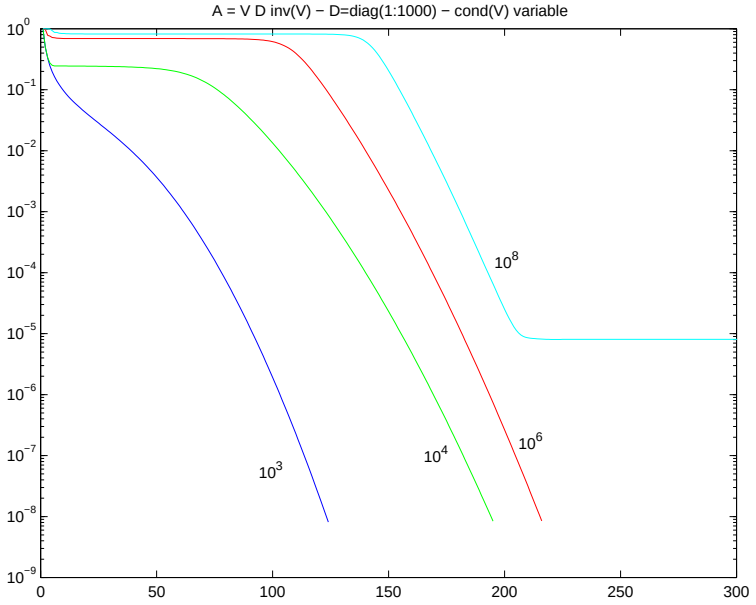
Risque de stagnation

GMRES - convergence

Valeurs propres variables



Vecteurs propres variables



GMRES - algorithme

Arnoldi

Rotations de Givens pour les moindres carrés

Redémarrage

GMRES - dépendances

dépendances dans Arnoldi
produit matrice creuse-vecteur
moindres carrés séquentiel

Arnoldi parallèle

découpler calcul de la base et orthogonalisation

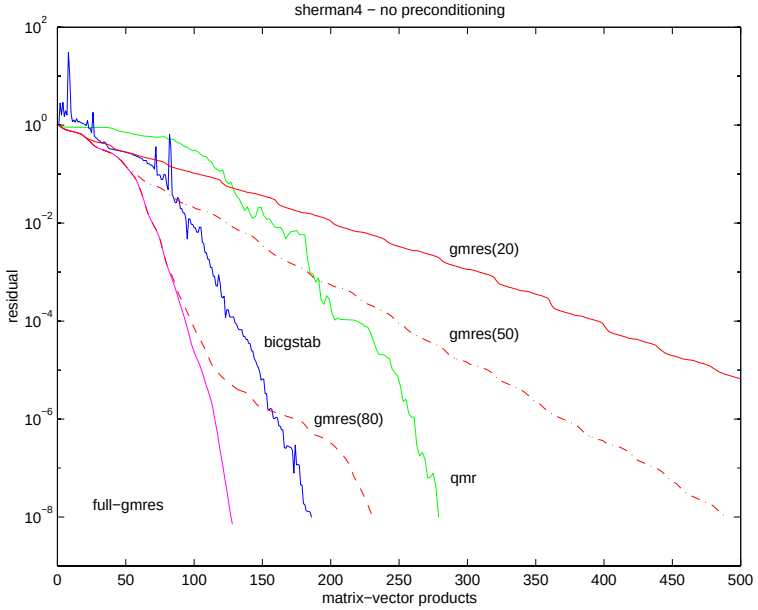
Comparaison

matrices Harwell-Boeing
matrix market

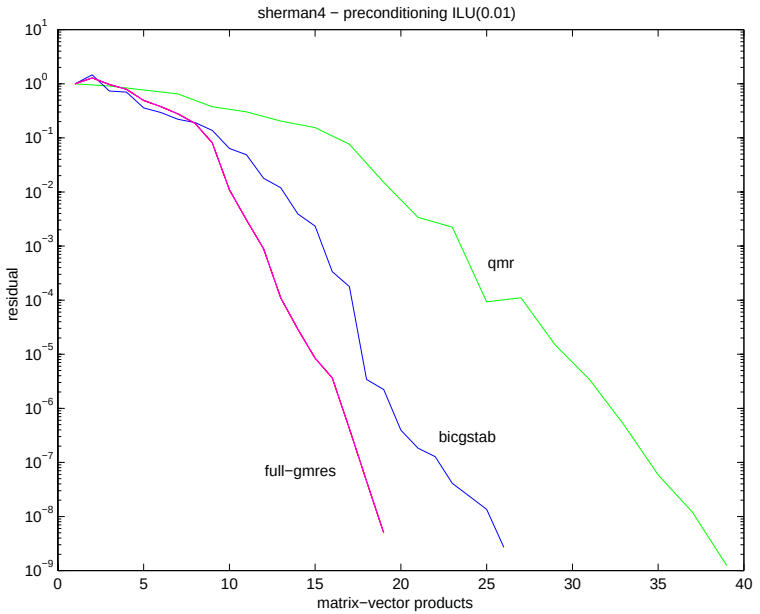
Comparaison - exemple

Matrice Sherman4 - n=1104 - nz=3786 - collection Boeing-Harwell

Sans préconditionnement



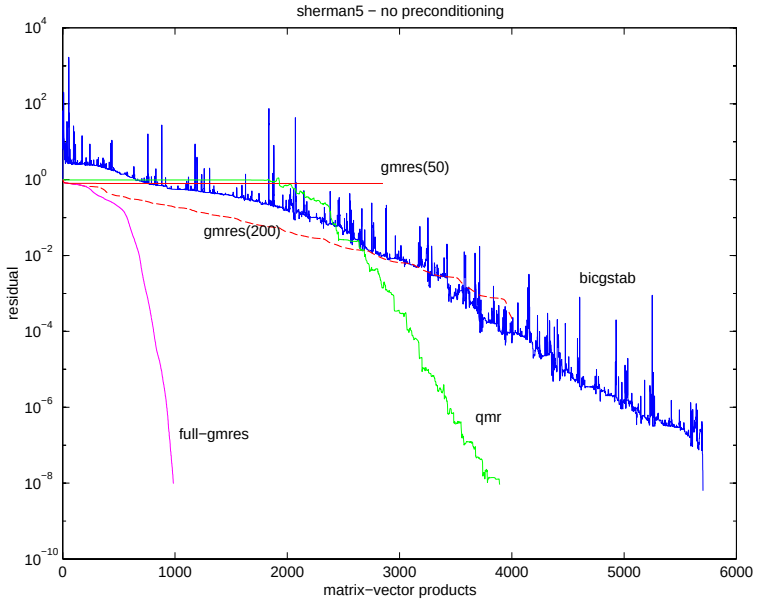
Avec ILU(0.01)



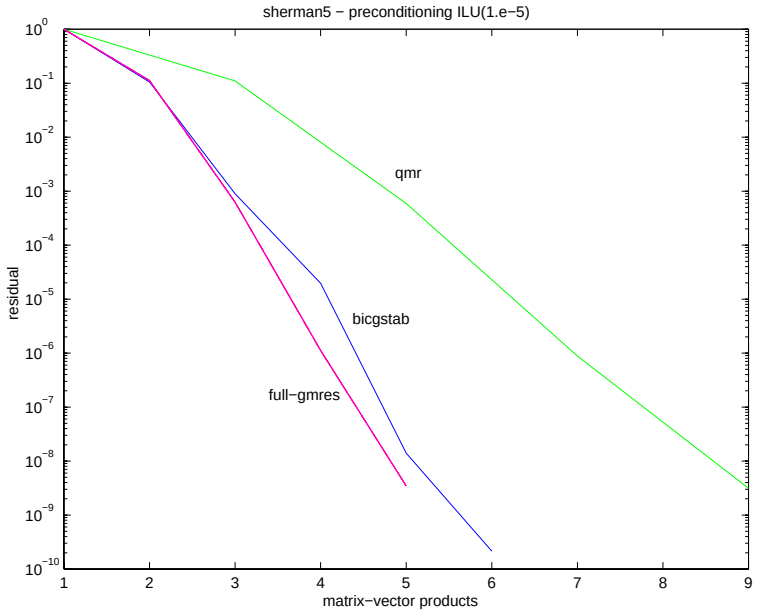
Comparaison - exemple

Matrice Sherman5 - n=3312 - nz=20793 - collection Boeing-Harwell

Sans préconditionnement



Avec ILU(0.01)



Préconditionnements

- Jacobi et SSOR
- Factorisations incomplètes
- Méthodes de Schwarz
- Méthodes hiérarchiques

Préconditionnements de Jacobi et SSOR

décomposition $A = D + L + U$,
D diagonale, L triangulaire inférieure, U triangulaire supérieure

Jacobi

$M = D$: parallèle mais peu efficace

SSOR

$M = (D + L)D^{-1}(D + U)$: plus efficace mais séquentiel
Si A est symétrique définie positive, M l'est aussi

Versions par blocs

D,L,U matrices diagonale et triangulaires par blocs
plus efficace

Préconditionnements par factorisation incomplète

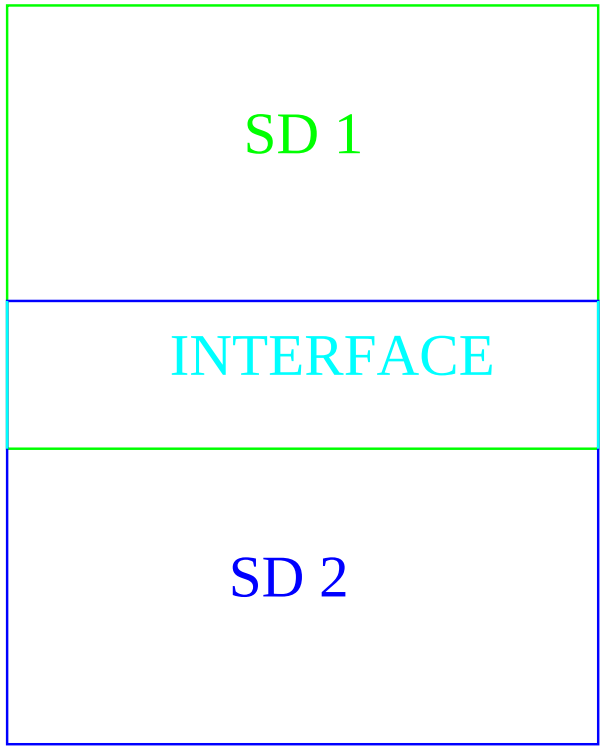
$A = LU + R$

différentes stratégies pour le choix de R

ILU(0) : pas de remplissage

Préconditionnements de Schwarz

Partition du domaine ou du graphe de A en sous-domaines avec recouvrement



p sous-domaines Ω_i
R_i restriction de Ω dans Ω_i
R_i^T extension de Ω_i dans Ω
A dans Ω_i : $A_i = R_i^T A R_i$
M_i préconditionnement de A_i

Préconditionnement de Schwarz multiplicatif

résolution de $Mz = r$

Algorithme

$r_1 = R_1 r$
$M_1 z_1 = r_1$
$z^{(1)} = R_1^T z_1$
$t = A z^{(1)}$
$t_2 = R_2 t$
$r_2 = R_2 r$
$M_2 z_2 = r_2 - t_2$
$z^{(2)} = R_2^T z_2$
$z = z^{(1)} + z^{(2)}$

Dépendances de données

$t_2 \neq 0$ si les deux sous-domaines se recouvrent
sous-domaine 2 après sous-domaine 1
addition globale dans z

$$M = R_1^T M_1 R_1 + R_2^T M_2 R_2 (I - A R_1^T M_1 R_1)$$

Préconditionnement de Schwarz additif

résolution de $Mz = r$

Algorithme

$r_1 = R_1 r$

$M_1 z_1 = r_1$

$z^{(1)} = R_1^T z_1$

$r_2 = R_2 r$

$M_2 z_2 = r_2$

$z^{(2)} = R_2^T z_2$

$z = z^{(1)} + z^{(2)}$

Dépendances de données

sous-domaine 2 en même temps que sous-domaine 1

addition globale dans z

$$M = R_1^T M_1 R_1 + R_2^T M_2 R_2$$

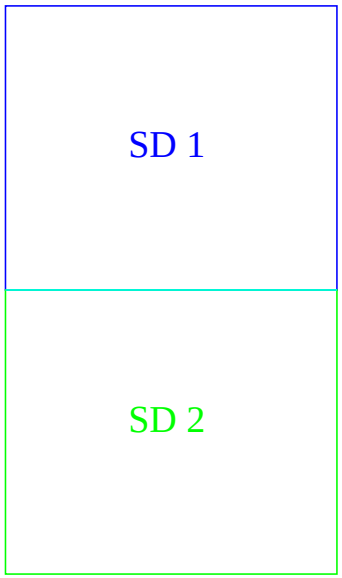
Correction par une grille grossière

Ω_0 ensemble des interfaces entre les sous-domaines
R_0 restriction et R_0^T extension
$A_0 = R_0^T A R_0$ et M_0 préconditionnement
correction $R_0^T M_0 R_0$
$M = R_1^T M_1 R_1 + R_2^T M_2 R_2 + R_0^T M_0 R_0$
correction effectuée en séquentiel

Nombre d'itérations indépendant du nombre de sous-domaines

Méthode de sous-domaines

Partition du domaine ou du graphe de A en sous-domaines



INTERFACE

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

$$\begin{cases} A_{11}x_1 = b_1 - A_{13}x_3 \\ A_{22}x_2 = b_2 - A_{23}x_3 \\ A_{33}x_3 = b_3 - A_{31}x_1 - A_{32}x_2 \end{cases}$$

Complément de Schur

Élimination de x_1 et de x_2 dans l'équation avec x_3

$$\begin{cases} x_1 = A_{11}^{-1}(b_1 - A_{13}x_3) \\ x_2 = A_{22}^{-1}(b_2 - A_{23}x_3) \\ (A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23})x_3 = b_3 - A_{31}A_{11}^{-1}b_1 - A_{32}A_{22}^{-1}b_2 \end{cases}$$

Distribution du complément de Schur

$$\begin{array}{ll} A_{33} = A_{33}^{(1)} + A_{33}^{(2)} & b_3 = b_3^{(1)} + b_3^{(2)} \\ S^{(1)} = A_{33}^{(1)} - A_{31}A_{11}^{-1}A_{13} & c^{(1)} = b_3^{(1)} - A_{31}A_{11}^{-1}b_1 \\ S^{(2)} = A_{33}^{(2)} - A_{32}A_{22}^{-1}A_{23} & c^{(2)} = b_3^{(2)} - A_{32}A_{22}^{-1}b_2 \\ S = S^{(1)} + S^{(2)} & c = c^{(1)} + c^{(2)} \end{array}$$

Système sur l'interface : $Sx = c$

Complément de Schur et Gradient Conjugué

Si A est symétrique définie positive, alors S l'est aussi.

Méthode du Gradient Conjugué pour résoudre $Sx = c$

A l'initialisation, calcul du second membre c
A chaque itération, produit matrice vecteur $y = Sx$
Préconditionnement $Mz = y$
Opérations vectorielles

Parallélisme sur chaque sous-domaine
Communications avec l'interface

Produit complément de Schur par vecteur

Complément de Schur

$$y = Sx = S^{(1)}x + S^{(2)}x$$

Produit $y^{(1)} = S^{(1)}x$

$$y = S^{(1)}x \Leftrightarrow \begin{pmatrix} 0 \\ y \end{pmatrix} = \begin{pmatrix} A_{11} & A_{13} \\ A_{31} & A_{33}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x \end{pmatrix}$$

Interprétation : problème de Dirichlet

Problème de Dirichlet sur chaque sous-domaine avec solution x fixée sur l'interface.
Calcul de la dérivée y de x sur l'interface.

Produit parallèle complément de Schur par vecteur

Sous-domaine 1	Interface	Sous-domaine 2
	Envoyer x	
Recevoir x		Recevoir x
Calculer $y^{(1)} = S^{(1)}x$		Calculer $y^{(2)} = S^{(2)}x$
Envoyer $y^{(1)}$		Envoyer $y^{(2)}$
	Recevoir $y^{(1)}$ et $y^{(2)}$	
	$y = Sx = y^{(1)} + y^{(2)}$	

Préconditionnement du complément de Schur

Préconditionnement dit de Neumann

$S = S^{(1)} + S^{(2)}$ d'où $S^{-1} \simeq S^{(1)-1} + S^{(2)-1}$
$M = S^{(1)-1} + S^{(2)-1}$

Résolution de $S^{(1)}x = y$

$$S^{(1)}x = y \Leftrightarrow \begin{pmatrix} A_{11} & A_{13} \\ A_{31} & A_{33}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x \end{pmatrix} = \begin{pmatrix} 0 \\ y \end{pmatrix}$$

Interprétation : Problème de Neumann

Problème de Neumann sur chaque sous-domaine avec dérivée y fixée à l'interface.
Calcul de la solution x à l'interface.

Préconditionnement de Neumann parallèle

Résoudre $Mx = y$

Sous-domaine 1	Interface	Sous-domaine 2
	Envoyer y	
Recevoir y		Recevoir y
Résoudre $S^{(1)}x^{(1)} = y$		Résoudre $S^{(2)}x^{(2)} = y$
Envoyer $x^{(1)}$		Envoyer $x^{(2)}$
	Recevoir $x^{(1)}$ et $x^{(2)}$	
	$x = x^{(1)} + x^{(2)}$	

Préconditionnement avec correction de grille grossière

Le nombre d'itérations du Gradient Conjugué augmente en général avec le conditionnement du système préconditionné.

Le conditionnement du système préconditionné augmente avec le nombre de sous-domaines.

Grâce à une correction de grille grossière, on obtient un système préconditionné avec un conditionnement, donc un nombre d'itérations en général, indépendant du nombre de sous-domaines.

La méthode parallèle passe à l'échelle.

Bibliographie et logiciels pour l'algèbre linéaire creuse

- Logiciels pour les méthodes directes
- Logiciels pour les méthodes itératives
- Bibliographie

Logiciels pour les méthodes directes

- SuperLU : http://www.cs.berkeley.edu/ demmel/SuperLU.html
  Auteurs J. Demmel et X. Li
- Matlab
- etc

Logiciels pour les méthodes itératives

- liste : http://www.netlib.org/utk/papers/iterative-survey/
- PETSc : http://www-fp.mcs.anl.gov/petsc/
  équations aux dérivées partielles,
  systèmes différentiels,
  systèmes non linéaires et linéaires,
  sur machine à mémoire distribuée
- Parpre : http://www.cs.utk.edu/ eijkhout/parpre.html
  préconditionnements parallèles pour méthodes itératives
  utilise Petsc, partie du projet Scalapack, auteurs V. Eijkhout et T. Chan
- Psparslib : http://www.cs.umn.edu/Research/arpa/p_sparslib/psp-abs.html
  méthodes itératives parallèles, auteurs Y. Saad et al.
- Matlab
- etc

Bibliographie

- Y. Saad, Iterative methods for sparse linear systems, PWS Publishing Company, 1996
  http://www-users.cs.umn.edu/ saad/books.html
- G. Meurant, Computer solution of large linear systems, North Holland, 1999
  http://perso.wanadoo.fr/gerard.meurant/#GACC
- Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition, R. Barrett et al., SIAM, 1994
  http://www.netlib.org/templates/index.html
- etc