Solution of linear systems with square sparse matrices

Yuri Vassilevski

Institute of Numerical Mathematics
Russian Academy of Science

The Rome-Moscow school of Matrix Methods and Applied Linear Algebra

18 October 2010, Moscow

Outline

- **Systems with sparse matrices**
- **Direct methods**
- **Iterative methods**
- **Concluding remarks**

Definitions

Let A be a square matrix of order n, $detA \neq 0$:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

How can we solve a linear system with A and a right hand side $b \in \Re^n$, i.e. find such $x \in \Re^n$ that

$$Ax = b?$$

Example of origin

- **Linear systems appear in many applications**

Example of origin

- **Linear systems appear in many applications**
- **We address matrices generated by discretizations of PDEs (major part of applications)**

Example of origin

- **Linear systems appear in many applications**
- **We address matrices generated by discretizations of PDEs (major part of applications)**
- **These matrices are sparse:** $nz = O(n)$

Example of origin

- **Linear systems appear in many applications**
- **We address matrices generated by discretizations of PDEs (major part of applications)**
- **These matrices are sparse:** $nz = O(n)$

Example: finite differences for $-u'' = f$ **in** $(0, 1)$**,** $u(0) = u(1) = 0$**.**

Example of origin

- **Linear systems appear in many applications**
- **We address matrices generated by discretizations of PDEs (major part of applications)**
- **These matrices are sparse:** $nz = O(n)$

Example: finite differences for $-u'' = f$ **in** $(0,1)$**,** $u(0) = u(1) = 0$**.**

$$A_1 = h^{-2} \begin{pmatrix} 2 & -1 & 0 & \dots \\ -1 & 2 & -1 & 0 \\ & \ddots & \ddots & \ddots \\ 0 & & & \\ \dots & 0 & -1 & 2 \end{pmatrix}$$

Meshes and spectral properties of matrices

Eigenvalue problem for A:

$$Aw = \lambda w$$

Meshes and spectral properties of matrices

Eigenvalue problem for A:

$$Aw = \lambda w$$

Assume that $A = A^T > 0$. Define $cond\, A = \frac{\lambda_{\max}}{\lambda_{\min}}$

Meshes and spectral properties of matrices

Eigenvalue problem for A:

$$Aw = \lambda w$$

Assume that $A = A^T > 0$. Define $cond\, A = \frac{\lambda_{\max}}{\lambda_{\min}}$

$$A_1: \qquad \lambda_{\min} = 4h^{-2}\sin^2(\pi h/2) < \pi^2, \quad \lambda_{\max} = 4h^{-2}\cos^2(\pi h/2) \approx 4h^{-2}$$

$$cond\, A_1 = O(h^{-2})$$

Meshes and spectral properties of matrices

Eigenvalue problem for A:

$$Aw = \lambda w$$

Assume that $A = A^T > 0$. Define $cond\, A = \frac{\lambda_{\max}}{\lambda_{\min}}$

$$A_1: \qquad \lambda_{\min} = 4h^{-2}\sin^2(\pi h/2) < \pi^2, \quad \lambda_{\max} = 4h^{-2}\cos^2(\pi h/2) \approx 4h^{-2}$$

$$cond\, A_1 = O(h^{-2})$$

Finite differences for $-\Delta u = f$ in $\Omega = (0,1)^2$, $u = 0$ on $\partial\Omega$:

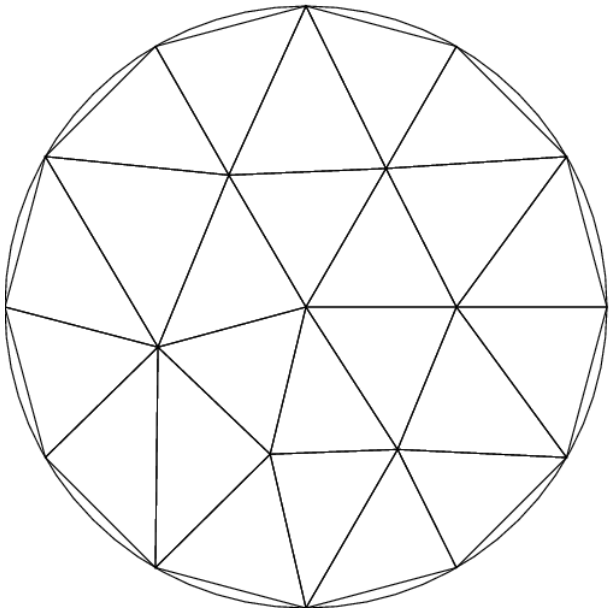
$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2$$

$$\lambda_{\min} = 8h^{-2}\sin^2(\pi h/2) < 2\pi^2, \quad \lambda_{\max} = 8h^{-2}\cos^2(\pi h/2) \approx 8h^{-2}$$

$$cond\, A_{12} = O(h^{-2})$$

Meshes and spectral properties of matrices

Finite elements for $-\Delta u = f$ in Ω, $u = 0$ on $\partial\Omega$:



quasiuniform mesh with mesh size h

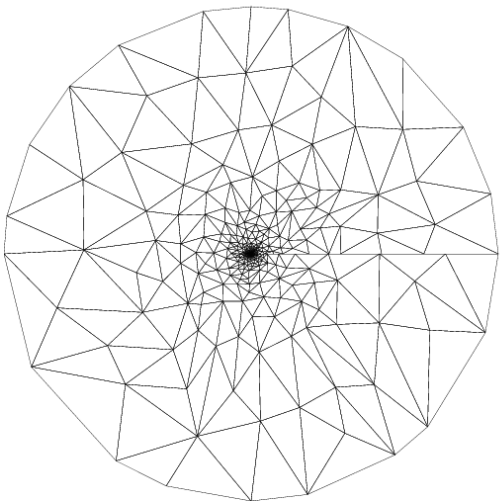
$$\lambda_{\min} = O(h^2),\ \lambda_{\max} = O(1)\ \rightarrow\ cond\, A = O(h^{-2})$$

Meshes and spectral properties of matrices

Finite elements for $-\Delta u = f$ in Ω, $u = 0$ on $\partial\Omega$:

$$\lambda_{\min} = O(h_{\max}^2), \ \lambda_{\max} = O(1)$$

$$cond\, A = O(h_{\max}^{-2})$$



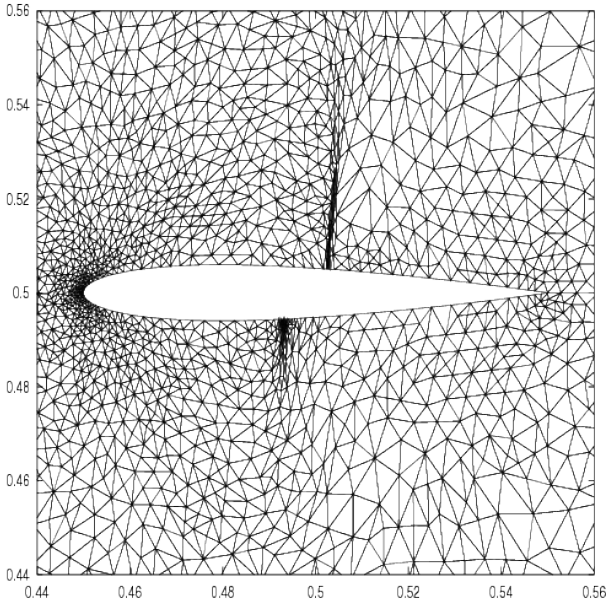
shape regular mesh with mesh size $[h_{\min}, h_{\max}]$

but may be

$$\lambda_{\min} = O(h_{\min}^2) \ \rightarrow \ cond\, A = O(h_{\min}^{-2})$$

Meshes and spectral properties of matrices

Finite elements for $-\Delta u = f$ in Ω , $u = 0$ on $\partial\Omega$:



adaptive anisotropic mesh

$$\lambda_{\min} = O((\alpha h)^2_{\min}),\ \lambda_{\max} = O(1)$$

$$cond\, A = O((\alpha h)^{-2}_{\min})$$

Classes of solvers

- **Strategy:**

Classes of solvers

- **Strategy:**
  - **Direct solvers**

Classes of solvers

- **Strategy:**
  - **Direct solvers**
  - **Iterative solvers**

Classes of solvers

- **Strategy:**
  - **Direct solvers**
  - **Iterative solvers**
- **Technology:**

Classes of solvers

- **Strategy:**
  - **Direct solvers**
  - **Iterative solvers**
- **Technology:**
  - **Black box**

Classes of solvers

- **Strategy:**
  - **Direct solvers**
  - **Iterative solvers**
- **Technology:**
  - **Black box**
  - **Gray box**

Classes of solvers

- **Strategy:**
  - **Direct solvers**
  - **Iterative solvers**
- **Technology:**
  - **Black box**
  - **Gray box**
  - **Problem dependent**

Sparse factorization

1. **Find permutations and factorize:**

$$P_1 A P_2 = LU$$

Sparse factorization

1. **Find permutations and factorize:**

$$P_1 A P_2 = LU$$

2. **Solve** $Ax = b$**:**

Sparse factorization

1. **Find permutations and factorize:**

$$P_1 A P_2 = LU$$

2. **Solve** $Ax = b$**:**
   - $Ly = P_1 b$

Sparse factorization

1. **Find permutations and factorize:**

$$P_1 A P_2 = LU$$

2. **Solve** $Ax = b$**:**
   - $Ly = P_1 b$
   - $Uz = y$

Sparse factorization

1. **Find permutations and factorize:**

$$P_1 A P_2 = LU$$

2. **Solve** $Ax = b$**:**
   - $Ly = P_1 b$
   - $Uz = y$
   - $P_2 x = z$

Sparse factorization

- **Advantages**

Sparse factorization

- **Advantages**
  - **Robust**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**
  - **Time consuming**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**
  - **Time consuming**
  - **Memory consuming**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**
  - **Time consuming**
  - **Memory consuming**
  - **Hard to parallelize**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**
  - **Time consuming**
  - **Memory consuming**
  - **Hard to parallelize**
  - **May be inefficient for systems of PDE's**

Sparse factorization

- **Advantages**
  - **Robust**
  - **Black box**
  - **Multiple rhs**
- **Drawbacks**
  - **Time consuming**
  - **Memory consuming**
  - **Hard to parallelize**
  - **May be inefficient for systems of PDE's**
- **Implementations: UMFPACK, MUMPS, WSMP, PARDISO, SuperLU**

Sparse factorization

2D Poisson problem

$$A_{12}x = b$$

$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2$$

$$nz_{LU} = O(n \ln n), \quad Flops = O(n^{\alpha}), \quad 1.5 \leq \alpha \leq 1.6$$

n_A	nz_A	MB	nz_{LU}	t_{factor}	t_{solve}
640000	**3196800**	**485**	$\mathbf{42 \cdot 10^6}$	**82**	**0.97**

Sparse factorization

3D Poisson problem

$$A_{123}x = b$$

$$A_{123} = A_1 \otimes I_2 \otimes I_3 + I_1 \otimes A_2 \otimes I_3 + I_1 \otimes I_2 \otimes A_3$$

$$nz_{LU} = O(n^{\alpha}), \quad 1.5 \leq \alpha \leq 1.6, \quad Flops = O(n^{\beta}), \quad 2.2 \leq \beta \leq 2.3$$

n_A	nz_A	MB	nz_{LU}	t_{factor}	t_{solve}
125000	**860000**	**926**	$\mathbf{82 \cdot 10^6}$	**1107**	**1.57**

Fast direct solver: Devide and conquer

Matrices from a special class

Two stages:

- **Initialization**
- **Solution**

Yuri Kuznetsov '84

$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2 + cI_1 \otimes I_2$$

rectangular non-uniform mesh

Fast direct solver: Devide and conquer

Matrices from a special class

Two stages:

- **Initialization**
- **Solution**

Yuri Kuznetsov '84

$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2 + cI_1 \otimes I_2$$

may be generalized to band matrices A_i, M_i

rectangular non-uniform mesh

$$A_{12} = A_1 \otimes M_2 + M_1 \otimes A_2 + cM_1 \otimes M_2$$

Fast direct solver: Devide and conquer

Matrices from a special class

Two stages:

- **Initialization**
- **Solution**

Yuri Kuznetsov '84

$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2 + cI_1 \otimes I_2$$

may be generalized to band matrices A_i, M_i

rectangular non-uniform mesh

$$A_{12} = A_1 \otimes M_2 + M_1 \otimes A_2 + cM_1 \otimes M_2$$

Complexity $Flops = O(n_1 n_2 \log_p n_1)$

Fast direct solver: Devide and conquer

Matrices from a special class

Two stages:

- **Initialization**
- **Solution**

Yuri Kuznetsov '84

$$A_{12} = A_1 \otimes I_2 + I_1 \otimes A_2 + cI_1 \otimes I_2$$

may be generalized to band matrices A_i, M_i

rectangular non-uniform mesh

$$A_{12} = A_1 \otimes M_2 + M_1 \otimes A_2 + cM_1 \otimes M_2$$

Complexity $Flops = O(n_1 n_2 \log_p n_1)$

Systems of order 10^8 are solved at laptops within 1 min!

Fast direct solver: Devide and conquer

Matrices from a special class

$$A_{123} = A_1 \otimes I_2 \otimes I_3 + I_1 \otimes A_2 \otimes I_3 + I_1 \otimes I_2 \otimes A_3 + cI_1 \otimes I_2 \otimes I_3$$

Complexity $Flops = O(n_1 n_2 n_3 \log_{p_1} n_1 \log_{p_3} n_3)$

Fast direct solver: Devide and conquer

Matrices from a special class

$$A_{123} = A_1 \otimes I_2 \otimes I_3 + I_1 \otimes A_2 \otimes I_3 + I_1 \otimes I_2 \otimes A_3 + cI_1 \otimes I_2 \otimes I_3$$

Complexity $Flops = O(n_1 n_2 n_3 \log_{p_1} n_1 \log_{p_3} n_3)$

Advantages:

- **Efficient**
- **Robust**
- **Parallelizable**

Abstract iterative solver

$$Ax = b$$

- **Choose an initial guess** x_0

Abstract iterative solver

$$Ax = b$$

- **Choose an initial guess** x_0
- **Do until convergence (**$\|Ax_k - b\| < \varepsilon$**):**

Abstract iterative solver

$$Ax = b$$

- **Choose an initial guess** x_0
- **Do until convergence (**$\|Ax_k - b\| < \varepsilon$**):**
- $x_k = F(x_{k-1}, \dots)$**,** $k = 1, 2, \dots$

Abstract iterative solver

$$Ax = b$$

- **Choose an initial guess** x_0
- **Do until convergence (**$\|Ax_k - b\| < \varepsilon$**):**
- $x_k = F(x_{k-1}, \dots)$**,** $k = 1, 2, \dots$
- $\dots$ **depend on the method**

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Decompose A in diagonal and strictly low and upper parts:

$$A = D - E - F$$

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Decompose A in diagonal and strictly low and upper parts:

$$A = D - E - F$$

$$Ax = b \quad \Leftrightarrow \quad (D - E - F)x = b$$

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Decompose A in diagonal and strictly low and upper parts:

$$A = D - E - F$$

$$Ax = b \quad \Leftrightarrow \quad (D - E - F)x = b$$

Jacobi iterations:

$$Dx_k = (E + F)x_{k-1} + b$$

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Decompose A in diagonal and strictly low and upper parts:

$$A = D - E - F$$

$$Ax = b \quad \Leftrightarrow \quad (D - E - F)x = b$$

Jacobi iterations:

$$Dx_k = (E + F)x_{k-1} + b$$

Gauss-Seidel iterations:

$$(D - E)x_k = Fx_{k-1} + b$$

Classical methods

Richardson's iterations:

$$x_k = x_{k-1} + \alpha(b - Ax_{k-1})$$

Decompose A in diagonal and strictly low and upper parts:

$$A = D - E - F$$

$$Ax = b \quad \Leftrightarrow \quad (D - E - F)x = b$$

Jacobi iterations:

$$Dx_k = (E + F)x_{k-1} + b$$

Gauss-Seidel iterations:

$$(D - E)x_k = Fx_{k-1} + b$$

In case of A_1 methods provide slow convergence, but fast frequencies are dumped quickly

$$\|e_k\| \leq (1 - ch^2)\|e_{k-1}\|, \quad e_k = x - x_k$$

Classical methods

Decompose A as

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)D)$$

Classical methods

Decompose A as

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)D)$$

SOR iterations:

$$(D - \omega E)x_k = (\omega F + (1 - \omega)D)x_{k-1} + \omega b$$

Classical methods

Decompose A as

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)D)$$

SOR iterations:

$$(D - \omega E)x_k = (\omega F + (1 - \omega)D)x_{k-1} + \omega b$$

SSOR:

$$(D - \omega E)x_{k-1/2} = (\omega F + (1 - \omega)D)x_{k-1} + \omega b$$
$$(D - \omega F)x_k = (\omega E + (1 - \omega)D)x_{k-1/2} + \omega b$$

Classical methods

Decompose A as

$$\omega A = (D - \omega E) - (\omega F + (1 - \omega)D)$$

SOR iterations:

$$(D - \omega E)x_k = (\omega F + (1 - \omega)D)x_{k-1} + \omega b$$

SSOR:

$$(D - \omega E)x_{k-1/2} = (\omega F + (1 - \omega)D)x_{k-1} + \omega b$$
$$(D - \omega F)x_k = (\omega E + (1 - \omega)D)x_{k-1/2} + \omega b$$

In case of A_1 methods provide better convergence if ω is optimal:

$$\|e_k\| \leq (1 - ch)\|e_{k-1}\|, \quad e_k = x - x_k$$

Krylov subspace methods

- $x_m - x_0 \in \mathcal{K}_m$, $\mathcal{K}_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, $r_0 = b - Ax_0$

Krylov subspace methods

- $x_m - x_0 \in \mathcal{K}_m$, $\mathcal{K}_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, $r_0 = b - Ax_0$
- **Basic operations:** Ax, (x, y), $y := ax + y$

Krylov subspace methods

- $x_m - x_0 \in \mathcal{K}_m$, $\mathcal{K}_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, $r_0 = b - Ax_0$
- **Basic operations:** Ax, (x, y), $y := ax + y$
- **Examples: CG (for $A = A^T > 0$), Lanczos (for $A = A^T$), GMRES**

Krylov subspace methods

- $x_m - x_0 \in \mathcal{K}_m$, $\mathcal{K}_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, $r_0 = b - Ax_0$
- **Basic operations:** Ax, (x, y), $y := ax + y$
- **Examples: CG (for $A = A^T > 0$), Lanczos (for $A = A^T$), GMRES**
- **Objectives: automatic choice of parameters and fast convergence**

Krylov subspace methods

- $x_m - x_0 \in \mathcal{K}_m$, $\mathcal{K}_m = span\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$, $r_0 = b - Ax_0$
- **Basic operations:** Ax, (x, y), $y := ax + y$
- **Examples: CG (for $A = A^T > 0$), Lanczos (for $A = A^T$), GMRES**
- **Objectives: automatic choice of parameters and fast convergence**
- **Convergence depends on spectral bounds. In case of A_1 and CG:**

$$\|e_k\|_A \leq 2(1 - ch)^k \|e_0\|_A$$

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

or

$$Ax = b \Rightarrow AM^{-1}y = b, \quad x = M^{-1}y$$

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

or

$$Ax = b \Rightarrow AM^{-1}y = b, \quad x = M^{-1}y$$

M **is chosen so that**

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

or

$$Ax = b \Rightarrow AM^{-1}y = b, \quad x = M^{-1}y$$

M is chosen so that

- **$M^{-1}A$ is closer to I or bounds of spectrum for $M^{-1}A$ are closer to 1**

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

or

$$Ax = b \Rightarrow AM^{-1}y = b, \quad x = M^{-1}y$$

M is chosen so that

- **$M^{-1}A$ is closer to I or bounds of spectrum for $M^{-1}A$ are closer to 1**
- **Operation $M^{-1}z$ is relatively cheap**

Preconditioning

$$Ax = b \Rightarrow M^{-1}Ax = M^{-1}b$$

or

$$Ax = b \Rightarrow AM^{-1}y = b, \quad x = M^{-1}y$$

M is chosen so that

- **$M^{-1}A$ is closer to I or bounds of spectrum for $M^{-1}A$ are closer to 1**
- **Operation $M^{-1}z$ is relatively cheap**
- **Krylov subspace methods may be preconditioned**

Classical preconditioners

$$A = D - E - F$$

- **Jacobi** $M = D$

Classical preconditioners

$$A = D - E - F$$

- **Jacobi** $M = D$
- **Symmetric Gauss-Seidel** $M = (D - E)D^{-1}(D - F)$

Classical preconditioners

$$A = D - E - F$$

- **Jacobi** $M = D$
- **Symmetric Gauss-Seidel** $M = (D - E)D^{-1}(D - F)$
- **Symmetric SOR** $M = (D - \omega E)D^{-1}(D - \omega F)$

ILU preconditioners

$$A \approx LU \Rightarrow M = LU \Rightarrow M^{-1} = U^{-1}L^{-1}$$

ILU preconditioners

$$A \approx LU \Rightarrow M = LU \Rightarrow M^{-1} = U^{-1}L^{-1}$$

- **Fill-in by position (sparsity of A or A^m)**

ILU preconditioners

$$A \approx LU \Rightarrow M = LU \Rightarrow M^{-1} = U^{-1}L^{-1}$$

- **Fill-in by position (sparsity of A or A^m)**
- **Fill-in by value (relative thresholds)**

ILU preconditioners

$$A \approx LU \Rightarrow M = LU \Rightarrow M^{-1} = U^{-1}L^{-1}$$

- **Fill-in by position (sparsity of A or A^m)**
- **Fill-in by value (relative thresholds)**

- **Example: ILU (I.Kaporin,98), Poisson equation in $(0,1)^2$**

mesh	n_A	nz_A	nz_{LU}	t_{factor}	t_{solve}	itr_{PCG}
$1t_{2h}$	**66049**	**460289**	$\mathbf{1.1 \cdot 10^6}$	**2.2**	**2.5**	**62**
$1t$	**263169**	**1838081**	$\mathbf{4.4 \cdot 10^6}$	**11.6**	**27.2**	**126**
3c	**229343**	**1605267**	$\mathbf{3.7 \cdot 10^6}$	**11.8**	**39.9**	**242**

ILU preconditioners

$$A \approx LU \Rightarrow M = LU \Rightarrow M^{-1} = U^{-1}L^{-1}$$

- **Fill-in by position (sparsity of A or A^m)**
- **Fill-in by value (relative thresholds)**

- **Example: ILU (I.Kaporin,98), Poisson equation in $(0,1)^2$**

mesh	n_A	nz_A	nz_{LU}	t_{factor}	t_{solve}	itr_{PCG}
$1t_{2h}$	**66049**	**460289**	$\mathbf{1.1 \cdot 10^6}$	**2.2**	**2.5**	**62**
$1t$	**263169**	**1838081**	$\mathbf{4.4 \cdot 10^6}$	**11.6**	**27.2**	**126**
3c	**229343**	**1605267**	$\mathbf{3.7 \cdot 10^6}$	**11.8**	**39.9**	**242**

dependence on h: [$1t_{2h}$,1t]
dependence on mesh structure: [1t,3c]

$nz_{LU} \sim n$ **for the same sparsity**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

  1. **Set $u_K = v$**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

1. **Set $u_K = v$**
2. **For $l = K, K-1, \ldots, 1$: $x_l = R_l u_l$, $u_{l-1} = P_l(u_l - A_l x_l)$,**

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

1. **Set** $u_K = v$
2. **For** $l = K, K-1, \ldots, 1$**:** $x_l = R_l u_l$**,** $u_{l-1} = P_l(u_l - A_l x_l)$,
3. $w_0 = A_0^{-1} u_0$

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

1. **Set** $u_K = v$
2. **For** $l = K, K-1, \ldots, 1$**:** $x_l = R_l u_l$**,** $u_{l-1} = P_l(u_l - A_l x_l)$,
3. $w_0 = A_0^{-1} u_0$
4. **For** $l = 1, 2, \ldots, K$**:** $y_l = x_l + I_l w_{l-1}$, $w_l = y_l + R_l(u_l - A_l y_l)$,

Multigrid preconditioner

- **Assume a sequence of K hierarchical meshes be given**
- **Idea: high frequency errors are dumped by a smoother, low frequency errors are dumped on coarser meshes**
- **Introduce matrix A_l and smoother (Jacobi) R_l on mesh l, interpolator P_l from mesh l to $l-1$, prolongator I_l from mesh $l-1$ to l.**
- **Evaluation of $M^{-1}v$:**

1. **Set** $u_K = v$
2. **For** $l = K, K-1, \ldots, 1$**:** $x_l = R_l u_l$**,** $u_{l-1} = P_l(u_l - A_l x_l)$,
3. $w_0 = A_0^{-1} u_0$
4. **For** $l = 1, 2, \ldots, K$**:** $y_l = x_l + I_l w_{l-1}$, $w_l = y_l + R_l(u_l - A_l y_l)$,
5. **Set** $M^{-1}v = w_K$

Multigrid preconditioner

- **Advantages:**

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**
  - **Parallel** $\pm$

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**
  - **Parallel** $\pm$
- **Drawbacks:**

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**
  - **Parallel** $\pm$
- **Drawbacks:**
  - **Needs a sequence of hierarchical grids**

Multigrid preconditioner

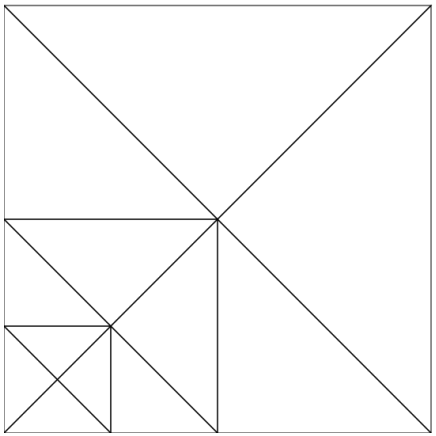
- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**
  - **Parallel** $\pm$
- **Drawbacks:**
  - **Needs a sequence of hierarchical grids**
  - **Needs definition of** A_l, R_l, I_l, P_l

Multigrid preconditioner

- **Advantages:**
  - **Complexity** $O(n)$
  - **Fast convergence**
  - **Parallel** $\pm$
- **Drawbacks:**
  - **Needs a sequence of hierarchical grids**
  - **Needs definition of** A_l, R_l, I_l, P_l
  - **Parallel** $\pm$

Multigrid preconditioner

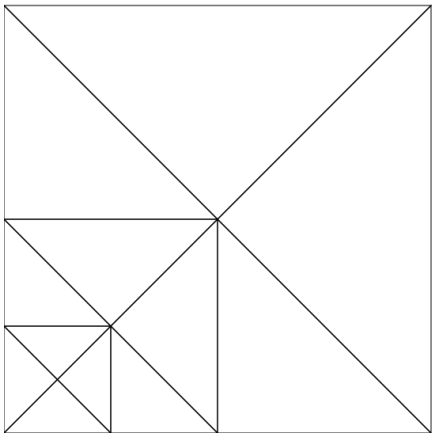
Extensions:



- **Locally refined meshes**

Multigrid preconditioner

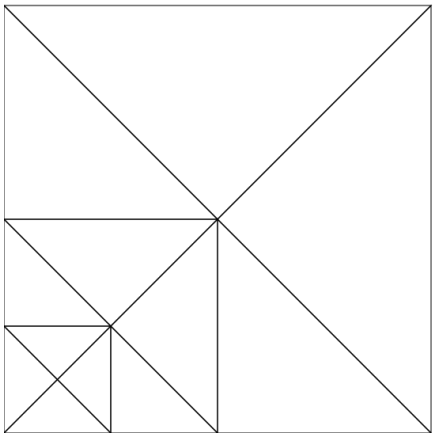
Extensions:



- **Locally refined meshes**
- **Multilevel methods (BPX)**

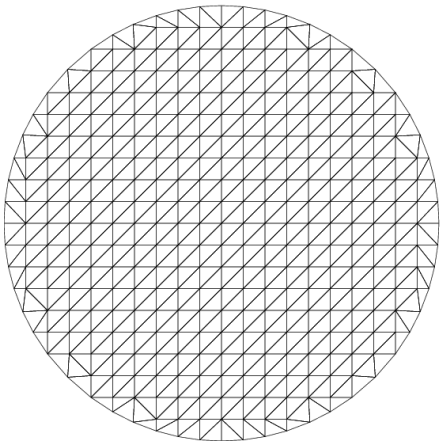
Multigrid preconditioner

Extensions:



- **Locally refined meshes**
- **Multilevel methods (BPX)**
- **Algebraic multigrids (black box)**

Fictitious domain preconditioner

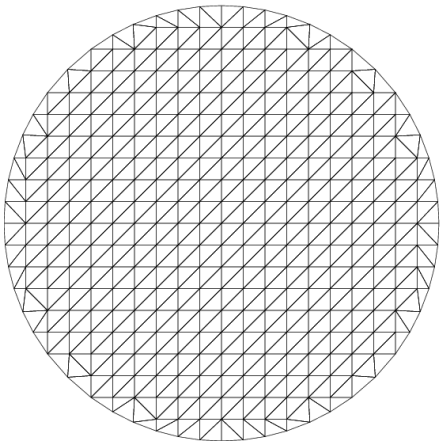


FEM discretization of Poisson equation with Neumann b.c.

Astrakhancev, Kuznetsov, Matsokin 70's

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_\Gamma \end{pmatrix}$$

Fictitious domain preconditioner



FEM discretization of Poisson equation with Neumann b.c.

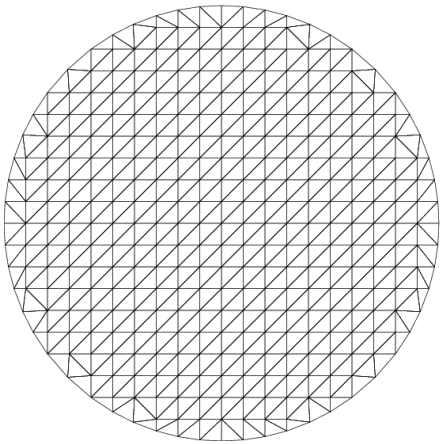
Astrakhancev, Kuznetsov, Matsokin 70's

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_\Gamma \end{pmatrix}$$

Consider extension B

$$B = \begin{pmatrix} A_{11} & A_{1\Gamma} & O \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} & O \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ O & A_{\Gamma}^{(2)} & A_{\Gamma 2} \\ O & A_{2\Gamma} & A_{22} \end{pmatrix}$$

Fictitious domain preconditioner



FEM discretization of Poisson equation with Neumann b.c.

Astrakhancev, Kuznetsov, Matsokin 70's

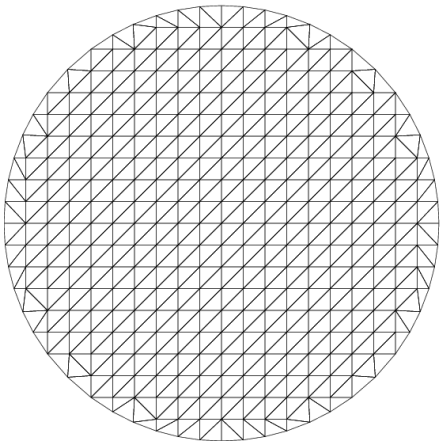
$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_\Gamma \end{pmatrix}$$

Consider extension B

$$B = \begin{pmatrix} A_{11} & A_{1\Gamma} & O \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} & O \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ O & A_{\Gamma}^{(2)} & A_{\Gamma 2} \\ O & A_{2\Gamma} & A_{22} \end{pmatrix}$$

and a preconditioner $\widetilde{B} \sim B$ on rectangular mesh (MG, D&C)

Fictitious domain preconditioner



FEM discretization of Poisson equation with Neumann b.c.

Astrakhancev, Kuznetsov, Matsokin 70's

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_\Gamma \end{pmatrix}$$

Consider extension B

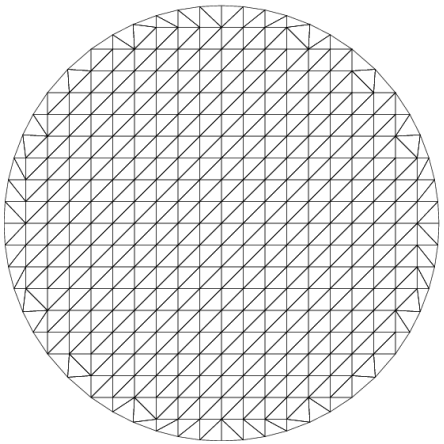
$$B = \begin{pmatrix} A_{11} & A_{1\Gamma} & O \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} & O \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ O & A_{\Gamma}^{(2)} & A_{\Gamma 2} \\ O & A_{2\Gamma} & A_{22} \end{pmatrix}$$

and a preconditioner $\widetilde{B} \sim B$ on rectangular mesh (MG, D&C)

Fictitious domain preconditioner

$$\begin{pmatrix} \widetilde{B}_{11} & \widetilde{B}_{1\Gamma} \\ \widetilde{B}_{\Gamma 1} & \widetilde{B}_{\Gamma} - \widetilde{B}_{\Gamma 2}\widetilde{B}_{22}^{-1}\widetilde{B}_{2\Gamma} \end{pmatrix}$$

Fictitious domain preconditioner



FEM discretization of Poisson equation with Neumann b.c.

Astrakhancev, Kuznetsov, Matsokin 70's

$$\begin{pmatrix} A_{11} & A_{1\Gamma} \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} x_1 \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_\Gamma \end{pmatrix}$$

Consider extension B

$$B = \begin{pmatrix} A_{11} & A_{1\Gamma} & O \\ A_{\Gamma 1} & A_{\Gamma}^{(1)} & O \\ O & O & O \end{pmatrix} + \begin{pmatrix} O & O & O \\ O & A_{\Gamma}^{(2)} & A_{\Gamma 2} \\ O & A_{2\Gamma} & A_{22} \end{pmatrix}$$

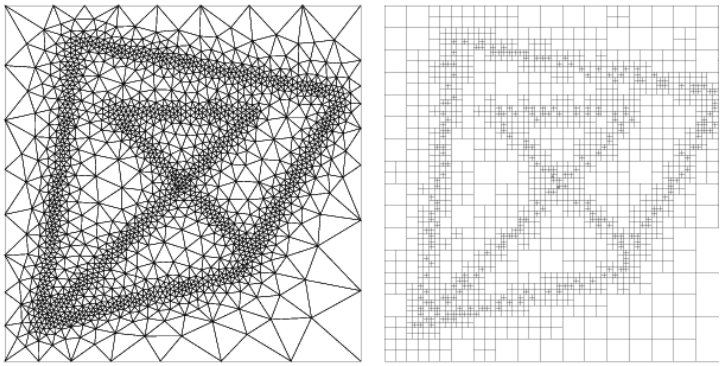
and a preconditioner $\widetilde{B} \sim B$ on rectangular mesh (MG, D&C)

Fictitious domain preconditioner

$$\begin{pmatrix} \widetilde{B}_{11} & \widetilde{B}_{1\Gamma} \\ \widetilde{B}_{\Gamma 1} & \widetilde{B}_{\Gamma} - \widetilde{B}_{\Gamma 2}\widetilde{B}_{22}^{-1}\widetilde{B}_{2\Gamma} \end{pmatrix}$$

fast convergence, parallel, partial solutions

Fictitious space preconditioner



Nepomnyaschikh ’91

$$M^{-1} = R\tilde{M}^{-1}R^{\star}$$

Domain decomposition preconditioner

- **Split computational domain Ω into m nonoverlapping subdomains Ω_i**

Domain decomposition preconditioner

- **Split computational domain Ω into m nonoverlapping subdomains Ω_i**
- **Interface between Ω_i is Γ**

Domain decomposition preconditioner

- **Split computational domain Ω into m nonoverlapping subdomains Ω_i**
- **Interface between Ω_i is Γ**
- **Decomposition of A into blocks**

$$A = \begin{pmatrix} A_{11} & O & \dots & A_{1\Gamma} \\ & \ddots & & \vdots \\ O & \dots & A_{mm} & A_{m\Gamma} \\ A_{\Gamma 1} & \dots & A_{\Gamma m} & A_{\Gamma} \end{pmatrix}$$

Domain decomposition preconditioner

- **Split computational domain Ω into m nonoverlapping subdomains Ω_i**
- **Interface between Ω_i is Γ**
- **Decomposition of A into blocks**

$$A = \begin{pmatrix} A_{11} & O & \dots & A_{1\Gamma} \\ & \ddots & & \vdots \\ O & \dots & A_{mm} & A_{m\Gamma} \\ A_{\Gamma 1} & \dots & A_{\Gamma m} & A_{\Gamma} \end{pmatrix}$$

- **Example of DD preconditioner:**

$$M = \begin{pmatrix} M_{11} & O & \dots & O \\ & \ddots & & \vdots \\ O & \dots & M_{mm} & O \\ O & \dots & O & M_{\Gamma} \end{pmatrix}$$

$$M_{\Gamma}^{-1} = \sum_{i=1}^{m} N_i D_i S_i^{-1} D_i N_i^T$$

$$\sum_{i=1}^{m} N_i D_i N_i^T = I, \qquad S_i = A_{\Gamma_i} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}$$

Domain decomposition preconditioner

- **Split computational domain Ω into m nonoverlapping subdomains Ω_i**
- **Interface between Ω_i is Γ**
- **Decomposition of A into blocks**

$$A = \begin{pmatrix} A_{11} & O & \dots & A_{1\Gamma} \\ & \ddots & & \vdots \\ O & \dots & A_{mm} & A_{m\Gamma} \\ A_{\Gamma 1} & \dots & A_{\Gamma m} & A_{\Gamma} \end{pmatrix}$$

- **Example of DD preconditioner:**

$$M = \begin{pmatrix} M_{11} & O & \dots & O \\ & \ddots & & \vdots \\ O & \dots & M_{mm} & O \\ O & \dots & O & M_{\Gamma} \end{pmatrix}$$

$$M_{\Gamma}^{-1} = \sum_{i=1}^{m} N_i D_i S_i^{-1} D_i N_i^T$$

$$\sum_{i=1}^{m} N_i D_i N_i^T = I, \qquad S_i = A_{\Gamma_i} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}$$

- **Fast convergence, parallel, only precondtioners, extension to large m (Mandel '93)**

Conclusions

- **Direct vs. iterative solvers**

Conclusions

- **Direct vs. iterative solvers**
- **Efficient and parallel problem dependent solvers (D&C, MG, DD, FD)**

Conclusions

- **Direct vs. iterative solvers**
- **Efficient and parallel problem dependent solvers (D&C, MG, DD, FD)**
- **Black/gray box solvers with restrictions (LU,ILU,AMG)**

Conclusions

- **Direct vs. iterative solvers**
- **Efficient and parallel problem dependent solvers (D&C, MG, DD, FD)**
- **Black/gray box solvers with restrictions (LU,ILU,AMG)**
- **Vast field of research**