Preconditioning Techniques

Matthias Bollhöfer (TU Braunschweig)

INRIA Sophia Antipolis
15. September 2010

Outline

Model Problems

1D elliptic boundary value problem

$$\begin{aligned} -u''(x) + \beta u'(x) + \gamma u(x) &= f(x),\ x \in [0,1] \\ u(0) &= g_0,\ u(1) = g_1 \end{aligned}$$

2D elliptic boundary value problem

$$\begin{aligned} \Omega &= [0,1] \times [0,1] \\ \overbrace{-u_{xx}(x,y) - u_{yy}(x,y)}^{-\Delta u(x,y)} + \beta u_x(x,y) + \gamma u(x,y) &= f(x,y),\ (x,y)\text{ß}n\Omega \\ u(x,y) &= g(x,y),\ (x,y) \in \partial\Omega \end{aligned}$$

3D elliptic boundary value problem

$$-\Delta u + \beta u_x + \gamma u = f \text{ in } [0,1]^3 + \text{b.c.}$$

Technische Universität Braunschweig

Discrete Linear System

- use discrete grid of mesh size $h = \frac{1}{n+1}$
- centered finite difference of second order for term $-\Delta u$
  $\longrightarrow$ stiffness matrix K_h

- first order upwind discretization for u_x
  $\longrightarrow$ matrix S_h

$$Au_h = f_h, \text{ where } A = K_h + \beta S_h + \gamma I$$

- $\beta = 0, \gamma \geqslant 0 \Rightarrow A = K_h + \gamma I$ is symmetric, and positive definite
  $\longrightarrow$ iterative solver CG can be used

- $\beta = 0, \gamma < 0 \Rightarrow A = K_h + \gamma I$ is symmetric, but indefinite
  $\longrightarrow$ iterative solver MINRES (or QMR for symmetric matrices) can be used

- $\beta \neq 0 \Rightarrow A = K_h + \beta S_h + \gamma I$ is unsymmetric
  $\longrightarrow$ general iterative solver (GMRES, BiCGstab,QMR,...) has to be used

Plain Iterative Solvers

The Positive Definite Case — CG

1D boundary value problem, $\beta = \gamma = 0$

h	problem size	comput. time[sec]	steps
$\frac{1}{31}$	31	0.002	16
$\frac{1}{63}$	63	0.004	32
$\frac{1}{127}$	127	0.007	64
$\frac{1}{255}$	255	0.010	128

Plain Iterative Solvers

The Positive Definite Case — CG

2D boundary value problem, $\beta = \gamma = 0$

h	problem size	comput. time[sec]	steps
$\frac{1}{31}$	961	0.01	60
$\frac{1}{63}$	3969	0.06	121
$\frac{1}{127}$	16129	0.47	230
$\frac{1}{255}$	65025	2.87	453

Plain Iterative Solvers

The Positive Definite Case — CG

3D boundary value problem, $\beta = \gamma = 0$

h	problem size	comput. time[sec]	steps
$\frac{1}{31}$	$3.0 \cdot 10^4$	0.3	79
$\frac{1}{63}$	$2.5 \cdot 10^6$	6.9	156
$\frac{1}{127}$	$2.0 \cdot 10^7$	103.4	294
$\frac{1}{255}$	$1.7 \cdot 10^8$	1638.2	579

Convergence theory:

$$\frac{\|x - x_l\|_A}{\|x - x_0\|_A} \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^l,$$

Since $\sqrt{\kappa} \approx \frac{2}{h\pi}$ for all spatial dimensions $\Rightarrow$ number of steps proportional to $\frac{1}{h}$

Introduction to Preconditioning

Objective in the SPD case:

- Compute $M \approx A^{-1}$ such that $\hat{\kappa} = \frac{\lambda_{\max}(AM)}{\lambda_{\min}(AM)} \ll \kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$

Good experience in the general case:

- Compute $M \approx A^{-1}$ such that $AM = I + E$, where
  - $\|E\|$ is small or
  - E is of low rank or
  - E has many eigenvalues close to 0.

Pros/Cons:

$\oplus$ SPD case: CG is expected to take less steps
general case: similar observation is often made

$\ominus$ $P = M^{-1}$ or M needs to computed

$\ominus$ application of AM instead of A is more expensive

Introduction to Preconditioning

Simple Examples

$$P = M^{-1}$$

1. choose $P = D$ as diagonal part of A (“Jacobi”)
2. choose $P = L$ as the lower triangular part of A (“Forward Gauss-Seidel”)
3. choose $P = U$ as the upper triangular part of A (“Backward Gauss-Seidel”)
4. choose $P = LD^{-1}U$ (“Symmetric Gauss-Seidel”)
5. block versions of 1–4
6. compute $A = LDU + R$, $P = LDU$ where some entries during Gaussian elimination are dropped (“ILU”)
7. compute M such that $\|AMe_i\|_2$ is small for $i = 1, \ldots, n$ (“SPAI”)

Plain Iterative Solvers

The General Case — restarted GMRES(30)

2D boundary value problem, $\beta = 1, \gamma = 0$

h	problem size	comput. time[sec]	steps
		JACOBI	
$\frac{1}{63}$	3969	2.3	520
$\frac{1}{127}$	16129	34.0	1581
$\frac{1}{255}$	65025	527.0	5435
		FORWARD GAUSS-SEIDEL	
$\frac{1}{63}$	3969	1.2	269
$\frac{1}{127}$	16129	15.4	710
$\frac{1}{255}$	65025	281.1	2894
		SYMMETRIC GAUSS-SEIDEL	
$\frac{1}{63}$	3969	0.4	97
$\frac{1}{127}$	16129	5.4	251
$\frac{1}{255}$	65025	73.2	724

Incomplete LU Factorization
ILU(0)

1 step LU:

$$A = \begin{pmatrix} \alpha & f^\top \\ e & C \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ e & I \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha} & 0 \\ 0 & S_C \end{pmatrix} \begin{pmatrix} \alpha & f^\top \\ 0 & I \end{pmatrix},$$

where $S_C = C - \frac{ef^\top}{\alpha}$ is the so-called Schur complement.
Repeat elimination step for S_C.

$$S_C = L_S D_S^{-1} U_S \Rightarrow A = \begin{pmatrix} \alpha & 0 \\ e & L_S \end{pmatrix} \begin{pmatrix} \alpha & 0 \\ 0 & D_S \end{pmatrix}^{-1} \begin{pmatrix} \alpha & f^\top \\ 0 & U_S \end{pmatrix},$$

Naive approximate decomposition, sparsify S_C:

$$S_C \longrightarrow \tilde{S}_C$$

where $\tilde{S}_C$ coincides with S_C, whereever C is nonzero.
This yields an approximate factorization of A.

$$A \approx \tilde{L}\tilde{D}^{-1}\tilde{U}.$$

"ILU(0)" (resp. IC(0) in the SPD case).

Incomplete LU Factorization

Strategies to suppress entries

- drop entries outside a specific pattern
- drop entries with higher level of fill
- drop entries with small modulus
- preserve structures (symmetry, SPD, diagonal dominance,. . .)

Efficient algorithms require appropriate data structures

In the sequel: A is stored in compressed row storage

$$A = \begin{pmatrix} 2 & -1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -7 & 0 & 6 \end{pmatrix}$$

row pointer	1	4	6	7	9

column indices	1	2	4	1	4	3	2	4
values	2	-1	1	2	1	-1	-7	6

Incomplete LU Factorization

Adapted (incomplete) *LU* decomposition for matrices *A* in CSR storage

- access by rows
- elimination inside a row from left to right

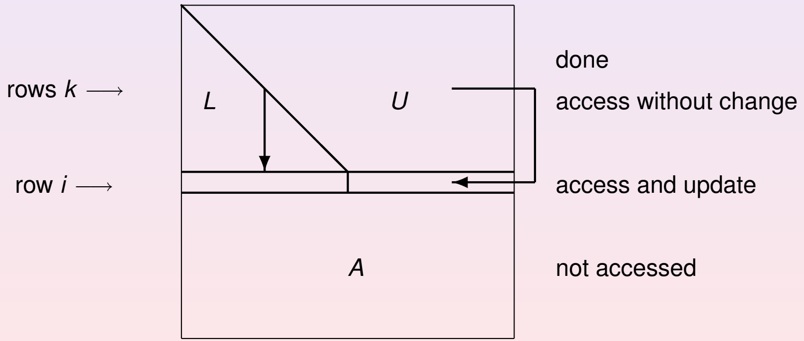
Sketch order of elimination

$$\left(\begin{array}{cccc} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{array}\right) \rightarrow \left(\begin{array}{cccc} * & * & * & * \\ \hline 0 & * & * & * \\ \hline * & * & * & * \\ * & * & * & * \end{array}\right) \rightarrow \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \\ \hline 0 & 0 & * & * \\ \hline * & * & * & * \end{array}\right) \rightarrow \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ \hline 0 & 0 & 0 & * \\ \hline \end{array}\right)$$

Incomplete LU Factorization

full LU factorization by rows, overwrite A by L, U

```
for $i = 2, \dots, n$:
    for $k = 1, \dots, i-1$:
        $a_{ik} := a_{ik} / a_{kk}$
        for $j = k+1, \dots, n$:
            $a_{ij} = a_{ij} - a_{ik} a_{kj}$
```



Incomplete LU Factorization

ILU(0) — straight forward implementation

```
for  $i = 2, \ldots, n$:
    for  $k = 1, \ldots, i-1$, whenever $a_{ik} \neq 0$
        $a_{ik} := a_{ik}/a_{kk}$
        for  $j = k+1, \ldots, n$, whenever $a_{ij} \neq 0$
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$
```

⊕ straight forward to implement.

⊖ often stability problems

$$\begin{pmatrix} 2 & -1 & 0 & 1 \\ 2 & 0 & 0 & 1 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ -8 & 0 & -1 & 2 \end{pmatrix}$$

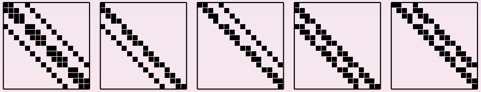
Incomplete LU Factorization

ILU(p) level-of-fill factorization

ILU(p), level *p* of fill–in depends on the kind of entry which produces the fill

Example ILU(1): Let $a_{ij}^{(0)}$ refer to the “original” level-0 entries

```
for   $i = 2, \dots, n$:
      for   $k = 1, \dots, i-1$, whenever $a_{ik} \neq 0$
            $a_{ik} := a_{ik} / a_{kk}$
              for   $j = k+1, \dots, n$, whenever $a_{ij} \neq 0$ or $a_{ik}^{(0)} a_{kj}^{(0)} \neq 0$
                    $a_{ij} = a_{ij} - a_{ik} a_{kj}$
```



$$\text{level}_{ij} = \min\{\text{level}_{ij}, \text{level}_{ik} + \text{level}_{kj} + 1\} \leqslant p$$

Incomplete LU Factorization

ILUT — threshold-based factorization

ILUT: drop entries with small value in modulus with respect to threshold $\tau \ll 1$.

```
for   $i = 1, \dots, n$:
      $w = (a_{i1}, \dots, a_{in})$, $\tau_i = \tau \|w\|$
      for   $k = 1, \dots, i-1$, whenever $w_k \neq 0$
            $w_k := a_{ik} / u_{kk}$. If $|w_k| \leqslant \tau_i$: $w_k = 0$

            If $w_k \neq 0$
                  for   $j = k+1, \dots, n$, whenever $u_{kj} \neq 0$
                        $w_j = w_j - w_k u_{kj}$
                  end
      end

      for   $j = 1, \dots, i-1$, whenever $w_j \neq 0$:   $l_{i,j} = w_j$

      for   $j = i, \dots, n$, whenever $w_j \neq 0$:   If $|w_j| > \tau_i$: $u_{i,j} = w_j$
      $w = 0$
```

work array w is managed similar to CSR format

Iterative Solvers with ILU Preconditioning

The General Case — BiCGstab

2D boundary value problem, $\beta = 1, \gamma = 0$

h	problem size	comput. time[sec]	steps
	ILU(0)		
$\frac{1}{127}$	16129	5.4	150
$\frac{1}{255}$	65025	85.1	273
	ILUT, $\tau = 10^{-2}$		
$\frac{1}{127}$	16129	0.3	67
$\frac{1}{255}$	65025	2.9	123
	ILUT, $\tau = 10^{-3}$		
$\frac{1}{127}$	16129	0.36	23
$\frac{1}{255}$	65025	3.5	41
	UMFPACK (MATLAB “\”)		
$\frac{1}{255}$	65025	1.0	

Iterative Solvers with ILU Preconditioning

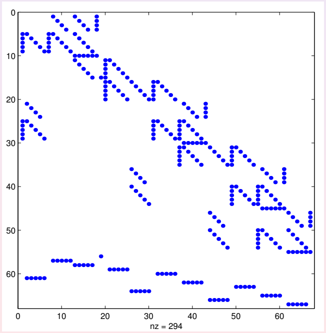
The General Case — BiCGstab

3D boundary value problem, $\beta = 1, \gamma = 0$

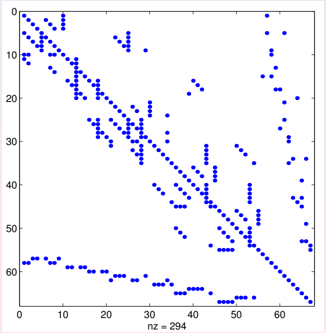
	ILUT, $\tau = 10^{-2}$		
$\frac{1}{31}$	$4.0 \cdot 10^4$	1.6	35
$\frac{1}{63}$	$2.5 \cdot 10^5$	60.5	57
	ILUT, $\tau = 10^{-3}$		
$\frac{1}{31}$	$4.0 \cdot 10^4$	5.9	17
$\frac{1}{63}$	$2.5 \cdot 10^5$	197.7	31
	UMFPACK (MATLAB "\")		
$\frac{1}{31}$	$4.0 \cdot 10^4$	2.8	
$\frac{1}{63}$	$2.5 \cdot 10^5$	156.0	

Incomplete LU Factorization

- in principle any of these ILUs could be supplemented with column pivoting easily
- additional preprocessing recommended
  - maximum weight matching (column permutation plus scaling) such that $|a_{i,\pi(i)}| = 1$, $|a_{i,\pi(j)}| \leqslant 1$ for all $i \neq j$
  - fill-reducing symmetric reorderings



$\longrightarrow$



Sparse Approximate Inverse Preconditioning — SPAI

$A = [a_1, \ldots, a_n]$, $M = (m_{ij})_{i,j} = [m_1, \ldots, m_n]$.
In any case we have

$$\min \|AM - I\|_F^2 = \min \sum_{j=1}^{n} \|Am_j - e_j\|_F^2 = \sum_{j=1}^{n} \left(\min \|Am_j - e_j\|_F^2 \right)$$

Formally, minimization can be done for every column of M separately!

Denote by $\mathcal{I}_j$ the nonzero pattern of M in column $j \Longrightarrow m_j = \sum_{i \in \mathcal{I}_j} m_{ij} e_j$.

$$\Longrightarrow \min \|Am_j - e_j\|_F^2 = \min \| \sum_{i \in \mathcal{I}_j} a_i m_{ij} - e_j \|_F^2$$

Solving this problem requires columns $(a_i)_{i \in \mathcal{I}_j}$ of A and computes m_j.

Sparse Approximate Inverse Preconditioning — SPAI

- suitable initial guess for pattern of M, e.g., I, A, $A^\top$ or $|A|+|A|^\top$
- we end up with least-squares problems of the following type

$$\|A_j x_j - e_j\|_2^2 = \min .$$

  These can be solved either with *QR*–decomposition or by solving the normal equations.

- pattern is adapted successively, e.g., using the residual $r = e_j - A_j x_j$.
  Choose additional column k such that $\|a_k m_{kj} - r\|_2^2$ is minimized.
  To do so, k has to be chosen s.t. $\frac{|r^\top a_k|}{\|a_k\|_2}$ is maximized.
  This only applies to a few columns, since r is initially sparse.

- Unfortunately: Often slow or M becomes dense!

Introduction to Multigrid
Smoothing Analysis

Why are elementary methods so bad?

For simplicity: 1D problem, $\beta = \gamma = 0$, Jacobi $D = \frac{2}{h^2} I$, basic iteration

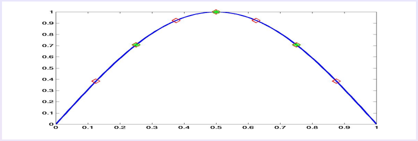
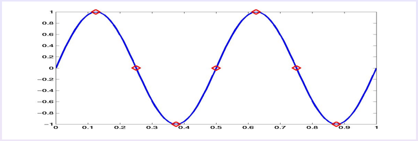
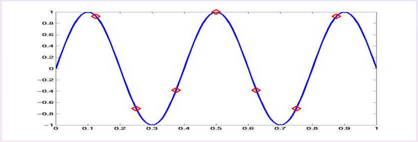
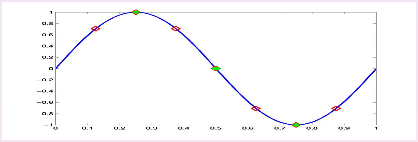
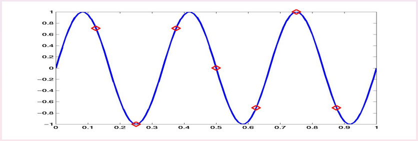
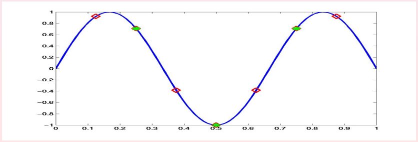
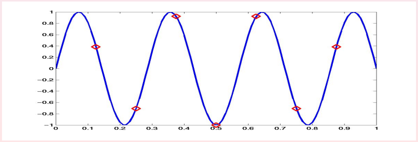
$$x^{(k+1)} = x^{(k)} + D^{-1}(b - A_h x^{(k)}).$$

error $e^{(k+1)} = x - x^{(k+1)}$ satisfies

$$e^{(k+1)} = x - x^{(k+1)} = x - \left[x^{(k)} + \frac{h^2}{2}(A_h x - A_h x^{(k)}) \right] = (I - \frac{h^2}{2} A_h) e^{(k)}$$

For the analysis use the eigenvectors $s^{(k)}$ and eigenvalues λ_k of A_h.

Smoothing Analysis — eigenvectors

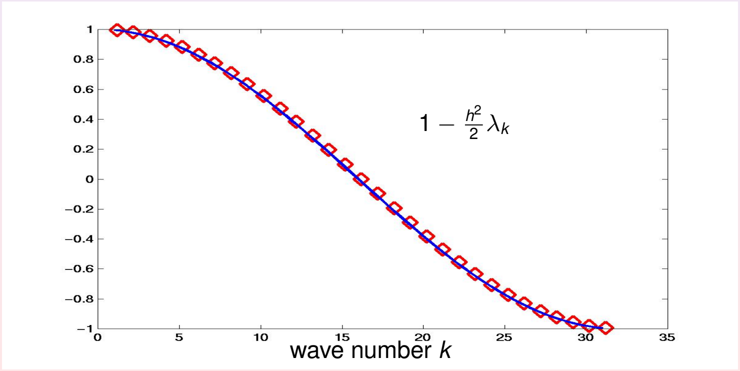








Smoothing Analysis — Jacobi method

Expand error e in terms of the eigenvectors as $e = \sum_{k=1}^{N} \alpha_k s^{(k)}$

$$(I - \frac{h^2}{2}A_h)e \quad = \quad \sum_{k=1}^{N} \alpha_k (s^{(k)} - \frac{h^2}{2}A_h s^{(k)}) = \sum_{k=1}^{N} \alpha_k \left(1 - \frac{h^2}{2}\lambda_k\right) s^{(k)}.$$

eigenvectors are damped by $\left|1 - \frac{h^2}{2}\lambda_k\right|$



Technische Universität Braunschweig

Smoothing Analysis — Jacobi method

Introduce damping parameter ω , i.e., $D \longrightarrow \frac{1}{\omega} D$

$$(I - \omega \frac{h^2}{2} A_h)e \quad = \quad \sum_{k=1}^{N} \alpha_k \left(1 - \omega \frac{h}{2} \lambda_k\right) s^{(k)}.$$

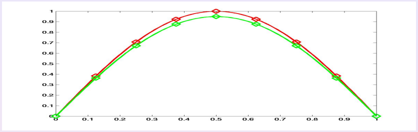
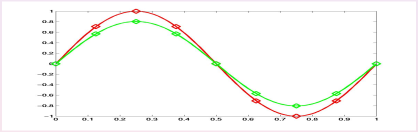
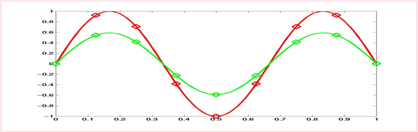
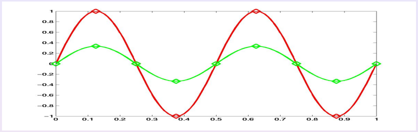
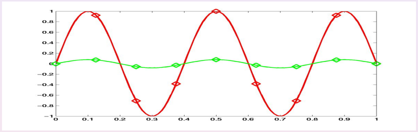
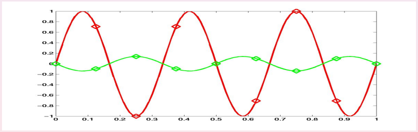
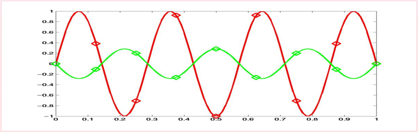
Optimize ω for $\left|1 - \omega \frac{h}{2} \lambda_k\right|$ for high frequencies $k \geqslant \frac{N}{2}$

$$\Longrightarrow \omega = \frac{2}{3}, \quad \left|1 - \frac{2}{3} \frac{h}{2} \lambda_k\right| \leqslant \frac{1}{3}, \text{ for all } k \geqslant \frac{N}{2}.$$

Technische Universität Braunschweig

Smoothing Analysis — Jacobi method

eigenvectors before and after smoothing

Smoothing Analysis — Observations

- undamped Jacobi method only damps frequencies in the medium range
- damped ($\omega = \frac{2}{3}$) Jacobi method damps high frequencies
- no damping for low frequencies

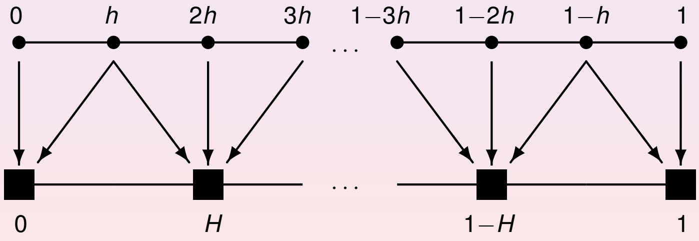
- BUT: low frequencies show up on the coarse grid with $H = 2h$ and A_{2h}
- We add a correction step to reduce the low frequencies.
  Idea: Use coarse grid Ω_{2h} and A_{2h}.

Coarse Grid Correction

$H = 2h$, transfer $\quad \begin{array}{ll} \Omega_h \longrightarrow \Omega_H & \text{restriction} \\ \Omega_H \longrightarrow \Omega_h & \text{interpolation} \end{array} \quad x \in \Omega_h,\ y \in \Omega_H$

restriction R. weighted average w.r.t. neighbours

$$y(iH) = \frac{1}{4}x(iH-h) + \frac{1}{2}x(iH) + \frac{1}{4}x(iH+h),\ \forall i$$



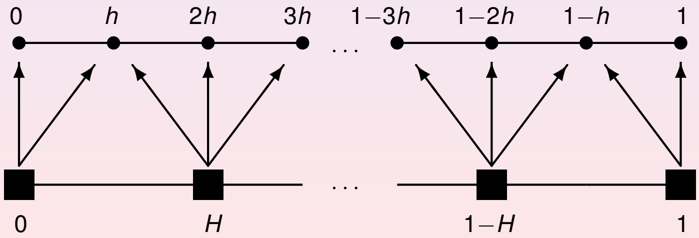
Coarse Grid Correction

$H = 2h$, transfer $\quad \Omega_h \longrightarrow \Omega_H$ restriction, $\Omega_H \longrightarrow \Omega_h$ interpolation $\quad x \in \Omega_h,\ y \in \Omega_H$

interpolation P. linear interpolation

$$x(ih) = \begin{cases} y(jH) & \text{if } i = 2j \\ \frac{1}{2}y(jH) + \frac{1}{2}y(jH+H) & \text{if } i = 2j+1 \end{cases}, \ \forall i$$

principle of the restriction being reversed. $P = 2R^\top$



Two Grid Method

Combine (damped) Jacobi with coarse grid correction $A_h^{-1} \approx PA_H^{-1}R$.

$$\begin{aligned} x_h^{(k+\frac{1}{2})} &= x_h^{(k)} + \omega D_h^{-1}(b_h - A_h x_h^{(k)}) \\ x_h^{(k+1)} &= x_h^{(k+\frac{1}{2})} + P_h A_H^{-1} R_h (b_h - A_h x_h^{(k+\frac{1}{2})}) \end{aligned}$$

Solving systems with A_H is recursively replaced by another instance of the two-grid method.

I.e., for $b_H = R_h(b_h - A_h x_h^{(k+\frac{1}{2})})$ replace $A_H x_H = b_H$ by

$$\left.\begin{aligned} x_H^{(\frac{1}{2})} &= x_H^{(0)} + \omega D_H^{-1}(b_H - A_H x_H^{(0)}) \\ x_H^{(1)} &= x_H^{(\frac{1}{2})} + P_H A_{2H}^{-1} R_H (b_H - A_H x_H^{(\frac{1}{2})}) \end{aligned}\right\} \Longrightarrow x_h^{(k+1)} \approx x_h^{(k+\frac{1}{2})} + P_h x_H^{(1)}$$

problem reduction $h \longrightarrow 2h = H \longrightarrow 2H = 4h \longrightarrow \cdots$

Multigrid

- smoothing analysis carries over two 2D, 3D with same ω for JACOBI
- interpolation/restriction are analogously defined in 2D/3D
- alternatively other smoothers $\mathcal{S}_h$ can be used, e.g. Gauss-Seidel without damping

- multigrid leads to
    - hierarchy of nested grids $\Omega_{h_1} \subseteq \Omega_{h_2} \subseteq \cdots \subseteq \Omega_{h_l}$, z.B. $h_{s+1} = h_s/2, \forall s$
    - sequence of discretized equations $A_{h_s} x_{h_s} = b_{h_s}$
    - sequence of interpolation operators P_{h_s} and restriction operators R_{h_s}

- number of smoothing steps $\nu = 1$ could be chosen also greater than 1
  $\rightarrow$ pre- and post- smoothing
- number of recursive calls $\mu = 1$ could be chosen also greater than 1
  $\rightarrow$ V- and W-cycle

Multigrid in 2D

2D boundary value problem, $\beta = \gamma = 0$

h	V–cycle, $\nu = 2$ iteration steps	W–cycle, $\nu = 2$ iteration steps
$\frac{1}{31}$	26	20
$\frac{1}{63}$	27	20
$\frac{1}{127}$	27	21
$\frac{1}{255}$	28	21

here multigrid is used with damped Jacobi smoothing

Multigrid Preconditioning

Any composed basic iterative method can be upgraded as preconditioner

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + B_1(b - Ax^{(k)}) \\ x^{(k+2)} &= x^{(k+1)} + B_2(b - Ax^{(k+1)}) \end{aligned}$$

$$\Rightarrow I - BA \equiv (I - B_2A)(I - B_1A) = I - [B_2 + B_1 - B_2AB_1]\, A$$

$$\Rightarrow \text{ preconditioner } M = B_1 + B_2 - B_2AB_1$$

2D boundary value problem with multigrid preconditioning and CG

	V–cycle, $\nu = 2$		W–cycle, $\nu = 2$	
h	damped Jacobi iteration steps	Gauss–Seidel iteration steps	damped Jacobi iteration steps	Gauss–Seidel iteration steps
$\frac{1}{31}$	11	9	10	8
$\frac{1}{63}$	11	9	10	8
$\frac{1}{127}$	12	9	10	8
$\frac{1}{255}$	12	9	10	8

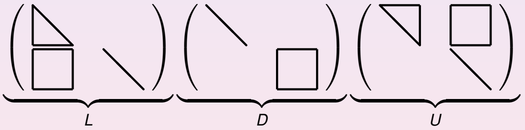
Multilevel ILU

Reordering (possibly rescaling) the system $\rightarrow \begin{cases} \mathcal{F} & \text{“fine grid nodes”} \\ \mathcal{C} & \text{“coarse grid nodes”} \end{cases}$

$$A \rightarrow \Pi^\top A \Pi = \begin{pmatrix} A_{\mathcal{F}\mathcal{F}} & A_{\mathcal{F}\mathcal{C}} \\ A_{\mathcal{C}\mathcal{F}} & A_{\mathcal{C}\mathcal{C}} \end{pmatrix}$$

Approximate block decomposition

$$\begin{pmatrix} A_{\mathcal{F}\mathcal{F}} & A_{\mathcal{F}\mathcal{C}} \\ A_{\mathcal{C}\mathcal{F}} & A_{\mathcal{C}\mathcal{C}} \end{pmatrix} = \begin{pmatrix} L_{\mathcal{F}\mathcal{F}} & 0 \\ L_{\mathcal{C}\mathcal{F}} & I \end{pmatrix} \begin{pmatrix} D_{\mathcal{F}\mathcal{F}} & 0 \\ 0 & S_{\mathcal{C}\mathcal{C}} \end{pmatrix} \begin{pmatrix} U_{\mathcal{F}\mathcal{F}} & U_{\mathcal{F}\mathcal{C}} \\ 0 & I \end{pmatrix} + E$$



$S_{\mathcal{C}\mathcal{C}}$ coarse grid matrix, E error matrix

- E arises from dropping entries of small size in L, U
- E might also arise from suppressing entries outside a specific pattern

Multilevel ILU

- Approximation $B_{\mathcal{FF}} \approx A_{\mathcal{FF}}^{-1}$, e.g. via solving with $L_{\mathcal{FF}} D_{\mathcal{FF}} U_{\mathcal{FF}}$
- $B_{\mathcal{FC}} \approx -A_{\mathcal{FF}}^{-1} A_{\mathcal{FC}}$, e.g. via $-L_{\mathcal{FF}}^{-1} L_{\mathcal{FC}}$
- $B_{\mathcal{CF}} \approx -A_{\mathcal{CF}} A_{\mathcal{FF}}^{-1}$, e.g. via $-U_{\mathcal{CF}} U_{\mathcal{FF}}^{-1}$

$$\begin{pmatrix} A_{\mathcal{FF}} & A_{\mathcal{FC}} \\ A_{\mathcal{CF}} & A_{\mathcal{CC}} \end{pmatrix}^{-1} \approx \begin{pmatrix} B_{\mathcal{FF}} & 0 \\ 0 & 0 \end{pmatrix} + \underbrace{\begin{pmatrix} B_{\mathcal{FC}} \\ I \end{pmatrix}}_{P} S_{\mathcal{CC}}^{-1} \underbrace{\begin{pmatrix} B_{\mathcal{CF}} & I \end{pmatrix}}_{R^\top}$$

- P “interpolation”
- $R^\top$ “restriction”

Multilevel approach: analogous principle recursively applied to $S_{\mathcal{CC}} = R^T A P$

Technische Universität Braunschweig

Multilevel ILU

- supplement with smoothing steps S_1, S_2 (e.g. Jacobi, Gauss–Seidel) iteration matrix for the amplified error $e = x - \tilde{x}$

$$e \to \left(I - \left\{ \begin{pmatrix} B_{\mathcal{F}\mathcal{F}} & 0 \\ 0 & 0 \end{pmatrix} + P S_{\mathcal{C}\mathcal{C}}^{-1} R^\top \right\} A\right) e$$

upgrade $\downarrow$ to Algebraic Multigrid

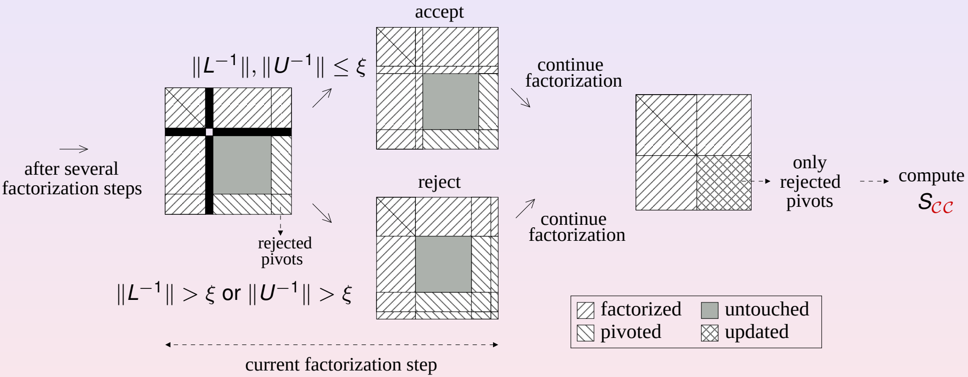
$$e \to (I - S_2 A)(I - P S_{\mathcal{C}\mathcal{C}}^{-1} R^\top A)(I - S_1 A) e$$

- V–cycle ($\mu = 1$), W–cycle ($\mu = 2$)

$$(I - S_2 A)(I - P S_{\mathcal{C}\mathcal{C}}^{-1} R^\top A)^{\mu}(I - S_1 A)$$

Inverse–Based Pivoting

- in principle we can estimate $\|L^{-1}\|, \|U^{-1}\|$ efficiently
- keep $\|L^{-1}\|, \|U^{-1}\|$ below ξ by inverse-based pivoting

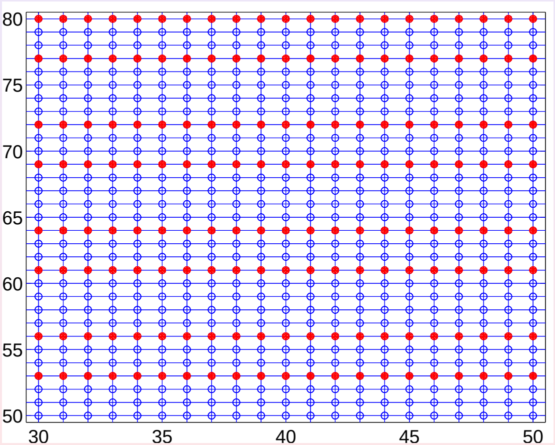


- inverse–based pivoting drives the coarsening process automatically
- $\longrightarrow$ postponed updates become the coarse grid system

Inverse–Based Pivoting

Example

$$\begin{aligned} -10^{-2}u_{xx}(x,y) - u_{yy}(x,y) &= f(x,y) \text{ for all } (x,y) \in [0,1]^2, \\ u(x,y) &= 0 \text{ on } \partial[0,1]^2 \end{aligned}$$



$*$ = rejected, $\circ$ = accepted

Inverse-based Pivoting

- Inverse–based pivoting directly yields

$$\|L^{-1}\|, \|U^{-1}\| \leqslant \xi$$

  for partial decomposition.
- If $\begin{pmatrix} A_{\mathcal{F}\mathcal{F}} & A_{\mathcal{F}\mathcal{C}} \\ A_{\mathcal{C}\mathcal{F}} & A_{\mathcal{C}\mathcal{C}} \end{pmatrix}$ has a large size block diagonal dominant block $A_{\mathcal{F}\mathcal{F}}$, then $\|L^{-1}\|$, $\|U^{-1}\|$ are small and a large portion of the system can be reduced.
- Inverse-based pivoting moves the low frequencies to $S_{\mathcal{C}\mathcal{C}}$

Inverse-based Pivoting

- Inverse–based pivoting directly yields

$$\|L^{-1}\|, \|U^{-1}\| \leqslant \xi$$

  for partial decomposition.

- If $\begin{pmatrix} A_{\mathcal{F}\mathcal{F}} & A_{\mathcal{F}\mathcal{C}} \\ A_{\mathcal{C}\mathcal{F}} & A_{\mathcal{C}\mathcal{C}} \end{pmatrix}$ has a large size block diagonal dominant block $A_{\mathcal{F}\mathcal{F}}$, then $\|L^{-1}\|$, $\|U^{-1}\|$ are small and a large portion of the system can be reduced.

- Inverse-based pivoting moves the low frequencies to $S_{\mathcal{C}\mathcal{C}}$

$$Ax = \varepsilon x$$

Inverse-based Pivoting

- Inverse–based pivoting directly yields

$$\|L^{-1}\|, \|U^{-1}\| \leqslant \xi$$

  for partial decomposition.

- If $\begin{pmatrix} A_{\mathcal{FF}} & A_{\mathcal{FC}} \\ A_{\mathcal{CF}} & A_{\mathcal{CC}} \end{pmatrix}$ has a large size block diagonal dominant block $A_{\mathcal{FF}}$, then $\|L^{-1}\|$, $\|U^{-1}\|$ are small and a large portion of the system can be reduced.

- Inverse-based pivoting moves the low frequencies to $S_{\mathcal{CC}}$

$$Ax = \varepsilon x$$

$$\underbrace{\frac{1}{\varepsilon}x}_{LARGE} = A^{-1}x \approx \begin{pmatrix} \underbrace{(L_{\mathcal{FF}} D_{\mathcal{FF}} L_{\mathcal{FF}}^T)^{-1}}_{\approx C} & 0 \\ 0 & 0 \end{pmatrix} x + \underbrace{P}_{\approx \xi} S_{\mathcal{CC}}^{-1} \underbrace{R}_{\approx \xi} x$$

Inverse-based Pivoting

- Inverse–based pivoting directly yields

$$\|L^{-1}\|, \|U^{-1}\| \leqslant \xi$$

for partial decomposition.

- If $\begin{pmatrix} A_{\mathcal{FF}} & A_{\mathcal{FC}} \\ A_{\mathcal{CF}} & A_{\mathcal{CC}} \end{pmatrix}$ has a large size block diagonal dominant block $A_{\mathcal{FF}}$, then $\|L^{-1}\|$, $\|U^{-1}\|$ are small and a large portion of the system can be reduced.

- Inverse-based pivoting moves the low frequencies to $S_{\mathcal{CC}}$

$$Ax = \varepsilon x$$

$$\underbrace{\frac{1}{\varepsilon} x}_{LARGE} = A^{-1} x \approx \begin{pmatrix} \underbrace{(L_{\mathcal{FF}} D_{\mathcal{FF}} L_{\mathcal{FF}}^T)^{-1}}_{\approx c} & 0 \\ 0 & 0 \end{pmatrix} x + \underbrace{P}_{\approx \xi} \underbrace{S_{\mathcal{CC}}^{-1}}_{LARGE} \underbrace{R}_{\approx \xi} x$$

$\Rightarrow$ by inverse-based pivoting $S_{\mathcal{CC}}$ captures the eigenvalues with small modulus

Multilevel ILU

Example

3D boundary value problem, $\beta = 1, \gamma = 0$

h	size	time ILU [sec]	$\frac{nnz(ILU)}{nnz(A)}$	time iteration [sec]	iteration steps
$\frac{1}{31}$	$3.0 \cdot 10^4$	0.7	3.2	0.4	20
$\frac{1}{63}$	$2.5 \cdot 10^5$	5.7	3.2	5.8	31
$\frac{1}{127}$	$2.0 \cdot 10^6$	59.3	3.6	128.4	61
$\frac{1}{255}$	$1.6 \cdot 10^7$	535.2	3.6	2215.5	124

Summary

- various collection of preconditioners
- significant improvement of Krylov subspace methods
- wherever possible, multigrid can be used as ’optimal’ preconditioner
- multilevel ILU closely connected
- coarse grid can be detected algebraically

Summary

- various collection of preconditioners
- significant improvement of Krylov subspace methods
- wherever possible, multigrid can be used as ’optimal’ preconditioner
- multilevel ILU closely connected
- coarse grid can be detected algebraically

Summary

- various collection of preconditioners
- significant improvement of Krylov subspace methods
- wherever possible, multigrid can be used as ’optimal’ preconditioner
- multilevel ILU closely connected
- coarse grid can be detected algebraically

This is the building where we are!