An Introduction to Preconditioners

Victor Eijkhout

594, March 2005
Introduction

Algebraic preconditioners
   Incomplete factorization (ILU) preconditioners
   Block methods
   Approximations of the inverse

Domain partitioning methods

Optimal solvers
   Multigrid
‘Simple’ preconditioners

Preconditioners constructed from matrix elements:

- **Jacobi**: \( M = D_A \)
- **Gauss-Seidel**: \( M = D_A + L_A \)
- **SOR**: \( M = D_A + \omega L_A \)
- **SSOR**: \( M = (\omega^{-1} D_A + L_A)((2\omega^{-1} - 1)D_A)^{-1} (\omega^{-1} D_A + U_A) \)
Classical theory of simple preconditioners

- Convergence condition:
  \[ \rho(I - M^{-1}A) < 1 \]

- Convergence guaranteed only for simple problems: M-matrices
- Jacobi and G-S: \#it \sim h^{-2}
- SOR: \#it \sim h^{-1} for optimal omega
Current state of simple preconditioners

- Stationary iterative methods are not used: convergence theory too limited
- Problem with G-S and SOR: nonsymmetry
- Only Jacobi still used with non-stationary methods, sometimes SSOR but only with $\omega = 1$
Preconditioners in non-stationary methods

- Convergence theory is incomplete: only bounds known for instance $\#it \sim \sqrt{\kappa(A)}$
- Possible criterium $\kappa(M^{-1}A) < \kappa(A)$ either order of magnitude or constant
Incomplete factorization (ILU) preconditioners
Direct methods: Gaussian elimination

\[ A = LU, \ Ax = b \implies x = A^{-1}b = U^{-1}(L^{-1}b) \]

- Problem with LU is fill-in: discarding fill-in gives approximate solution
- Aim: let \( LU \) take storage similar to \( A \)
Discarding fill-in

- Exact factorization:

  \[ \forall i, j > k : a_{ij} \leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}. \]

- Approximate factorization:

  \[ \forall i, j > k : \text{if } (i, j) \in S \quad a_{ij} \leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj}. \]

- Limit storage by defining \( S \)

- Alternatively: only fill in zero locations if \( a_{ik} a_{kk}^{-1} a_{kj} \) large enough
Error analysis of ILU

Laplacian:

\[ A = \begin{pmatrix}
4 & -1 & -1 \\
-1 & 4 & -1 \\
\vdots & \vdots & \vdots \\
-1 & 4 & -1 \\
\vdots & \vdots & \vdots \\
-1 & -1 & 4 & -1 & -1 \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix} \]

- One elimination step:

\[ A = \begin{pmatrix}
4 & -1 & -1 \\
3.75 & -1 & -0.25 & -1 \\
-1 & 4 & -1 \\
\end{pmatrix} \]

- One more elimination step:

\[ A = \begin{pmatrix}
4 & -1 & -1 \\
3.75 & -1 & -0.25 & -1 \\
3.73 & -1 & -0.66 & -2.666 & -1 \\
\end{pmatrix} \]

So, ILU(0) has an error of \( 1/p \) (typical pivot); ILU(1) has error \( 1/p^2 \), et cetera

594 Dongarra/Eijkhout 2005/03/16
Limit analysis

Because of the sign pattern, pivots decrease

⇒ inverses increase

⇒ limit pivot wanted
One-dimensional case:

\[
A = \begin{pmatrix}
2 & -1 & & \\
-1 & 2 & -1 & \\
-1 & 2 & -1 & \\
& & & \ddots
\end{pmatrix}
\]

- Pivot equation \(d_{i+1} = 2 - 1/d_i\)
- Limit equation \(d = 2 - 1/d\), solution \(d = 1\)

Two-dimensional case, pivot is altered by previous point and previous line:
\(d_{ij} = 4 - 1/d_{i-1,j} - 1/d_{i,j-1}\)
- Limit equation \(d = 4 - 2/d\), solution \(2 + \sqrt{2}\)
Fourier analysis

- Pivots converge quickly $\Rightarrow$ pretend they are constant
- Matrix and factorization have constant diagonal
- $\Rightarrow$ difference operators, eigenfunctions are sines
Modified ILU

- Instead of discarding fill-in, add to the diagonal

\[
\begin{align*}
    a_{ij} &\leftarrow a_{ij} - a_{ik} a_{kk}^{-1} a_{kj} \quad \text{accept fill} \\
    a_{ii} &\leftarrow a_{ii} - a_{ik} a_{kk}^{-1} a_{kj} \quad \text{discard fill}
\end{align*}
\]

- Physical meaning: conservation of mass

- Theory: possible reduction $\kappa(M^{-1}A) = O(h^{-1})$
If the only fill-in is on the diagonal, reuse storage of $L_A$ and $U_A$: only one extra vector for the preconditioner.

ILU-D: only allow fill-in on the diagonal

Lemma: ILU-D $\equiv$ ILU(0) if there are no triangles in the matrix graph

Proof: homework
Assume ILU-D, so \( L = L_A, U = U_A \). Different ways of writing the factorization:

\[
M = (D + L)D^{-1}(D + U) \\
= (I + LD^{-1})(D + U) \\
= (D + L)(I + D^{-1}U) \\
= (I + LD^{-1})D(I + D^{-1}U)
\]

- **Computational cost?**
- (2) and (3) cheaper if scaled factors are stored; otherwise all the same if we store \( D \) or \( D^{-1} \) as needed
- **Storage cost?**
- (1) and (4): two vectors; (2) and (3) only one
- ease of implementation?
Solving \((D + L)x = y\) or \((D + U)x = y\) simple:

\[
x_i \leftarrow d_i^{-1}(y_i - \sum_{j<i} \ell_{ij}x_j) \quad i = 1 \ldots n
\]

or

\[
x_i \leftarrow d_i^{-1}(y_i - \sum_{j>i} u_{ij}x_j) \quad i = 1 \ldots n
\]

However, \((I + D^{-1}U)x = y\)

\[
x_i \leftarrow y_i - d_i^{-1} \sum_{j>i} u_{ij}x_j
\]

while \((I + LD^{-1})x = y\)

\[
x_i \leftarrow y_i - \sum_{j<i} d_j^{-1} \ell_{ij}x_j
\]
Permutations of a matrix are allowed: Permuting for mathematical properties, or vectorization / parallelization

- Colour: set of points that are uncoupled;
- Colouring: division of the variables into sets of colours
- Parallelism: local solves

\[ x_i \leftarrow d_i^{-1}(b_i - \sum_{j \neq i} a_{ij}x_j) \]

are independent

- Matrix structure from colouring: diagonal diagonal blocks
Colouring theory

Theory: finding smallest number of colours is NP-complete, but not necessary; heuristics are allowed; parallelism desirable
Jones-Plassman colouring

- Assign random value to each node
- Find nodes with higher value than neighbours; these are uncouple
  \[ \Rightarrow \text{ Colour 1} \]
- Find nodes with a higher value than neighbours except colour one: again uncoupled; colour two
- Et cetera. This is largely parallel; does not give optimal number of colours, but close.
- Show that Modified ILU(0) on a red-black ordered domain leads to zero pivots.
Block methods
Factorization by subblocks

- Coupled differential equations: matrix has small number of large blocks
- Different numbering: each element is a small square block
- Physical: factorization by lines or planes

five-point Laplacian: \[ A = \begin{pmatrix} D & U \\ L & D & U \\ & \ddots & \ddots & \ddots \end{pmatrix}, \]

with \[ D = \begin{pmatrix} 4 & -1 \\ -1 & 4 & -1 \\ & \ddots & \ddots & \ddots \end{pmatrix}, \quad U = L = -I. \]
Gaussian elimination:

\[ A_{ij} \leftarrow A_{ij} - A_{ik} A^{-1}_{kk} A_{kj} \]

Problem: \( A_{kk} \) is matrix, so inverse needed

Case of small blocks: exact inverse

Larger blocks: approximate inverse
Basic idea: let $M \approx A^{-1}$, then explicit operation $x \leftarrow My$

- Factorizations are inherently recursive: Hard in parallel
- Explicit operations are very parallel
- Con: no geometric decay in the inverse
Approximations from factorization

- Series expansion: $(I - L)^{-1} = I + L + L^2 + L^3 + \cdots$
- Convergence?
- Number of terms is $N$: too large;
  convergence if matrix diagonally dominant
- Other expansion: $(I - L)^{-1} = (I + L)(I + L^2)(I + L^4)\cdots$
- Efficiency?
- Construction and application are vector/parallel
Approximation by minimization

- Determine a sparsity pattern $S$
- Minimize $\|I - MA\|$ where $M$ has nonzeros in $S$
- Parallel construction!
- Intrinsic problem: inverse may not have decay; choice of right sparsity pattern is hard.
Factored approximations

- Factorization of $A^{-1} = LU$
- Possible to compute banded parts of $L$ and $U$
Basic idea

- Partitioning into physical subdomains
- Elliptic problem on whole domain $\Rightarrow$ elliptic problem on subdomain
- Automatic parallelism of subdomains
- Use existing methods on subdomains
- Interesting theoretical properties
Connections between subdomains

- Zero overlap: block jacobi
- Positive overlap: Scharz method
- Negative overlap: Schur complement methods
Block Jacobi

- Very easy to program
- No parallel communication in the preconditioner
- Condition number improves by a constant
Originally invented for theoretical purposes
Schwarz methods

- Additive Schwarz: sum contributions on overlap
- Multiplicative Schwarz: overwrite contributions on overlap
- Can be optimal, may need global component
Schur complement methods

- Explicit interface; new linear system on interface

\[
\begin{pmatrix}
A_{11} & A_{13} \\
A_{22} & A_{23} \\
A_{31} & A_{32} \\
\end{pmatrix}
\Rightarrow
S = A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23}
\]

- Condition $O(h^{-1})$ on interface
- Trouble distributing the interface system
Local methods

- Matrix split in local/remote part
- Domain decomposition methods require solving local part
- Use ILU, SPAI,…
Optimal solvers
Spectral equivalence

Definition:

\[ c_1 x^t A^x \leq x^t M^{-1} A x \leq c_2 x^t A x \]

with \( c_1, c_2 \) independent of matrix size.

Then \( \kappa(M^{-1} A) = c_2 / c_1 \), so number of iterations is \( O(1) \).

Practical use: Laplace as preconditioner, solved by FFT, Recursive Bisection, et cetera.
Multigrid
On a grid of $n$ points, $\sin(n + m)\pi x$ and $\sin(n - m)\pi x$ ‘look the same’

Now combine two facts:

1. Low frequencies are hardest to solve
2. Low frequencies become high frequencies on a coarser grid
Multigrid algorithm

- Use a simple method (Gauss-Seidel) for just a few iterations
- Restrict current solution to coarser grid
- Solve there
- Interpolate back
- Do a bit more Gauss-Seidel

And do this recursively, then possibly optimal