

Lecture 22

Preconditioning, Incomplete Factorizations

MIT 18.335J / 6.337J

Introduction to Numerical Methods

Per-Olof Persson

November 30, 2006

Preconditioners for Linear Systems

- Main idea: Instead of solving

$$Ax = b$$

solve, using a nonsingular $m \times m$ preconditioner M ,

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

which has the same solution x

- Convergence properties based on $M^{-1}A$ instead of A
- Trade-off between the cost of applying M^{-1} and the improvement of the convergence properties. Extreme cases:
 - $M = A$, perfect conditioning of $M^{-1}A = I$, but expensive M^{-1}
 - $M = I$, “do nothing” $M^{-1} = I$, but no improvement of $M^{-1}A = A$

Preconditioned Conjugate Gradients

- To keep symmetry, solve $(C^{-1}AC^{-*})C^*x = C^{-1}b$ with $CC^* = M$
- Can be written in terms of M^{-1} only, without reference to C :

Algorithm: Preconditioned Conjugate Gradients Method

$$x_0 = 0, r_0 = b, p_0 = M^{-1}r_0, z_0 = p_0$$

for $n = 1, 2, 3, \dots$

$$\alpha_n = (r_{n-1}^T z_{n-1}) / (p_{n-1}^T A p_{n-1})$$

step length

$$x_n = x_{n-1} + \alpha_n p_{n-1}$$

approximate solution

$$r_n = r_{n-1} - \alpha_n A p_{n-1}$$

residual

$$z_n = M^{-1}r_n$$

preconditioning

$$\beta_n = (r_n^T z_n) / (r_{n-1}^T z_{n-1})$$

improvement this step

$$p_n = z_n + \beta_n p_{n-1}$$

search direction

Commonly Used Preconditioners

- A preconditioner should “approximately solve” the problem $Ax = b$
- **Jacobi preconditioning** - $M = \text{diag}(A)$, very simple and cheap, might improve certain problems but usually insufficient
- **Block-Jacobi preconditioning** - Use block-diagonal instead of diagonal. Another variant is using several diagonals (e.g. tridiagonal)
- **Classical iterative methods** - Precondition by applying one step of Jacobi, Gauss-Seidel, SOR, or SSOR
- **Incomplete factorizations** - Perform Gaussian elimination but ignore fill, results in approximate factors $A \approx LU$ or $A \approx R^T R$ (more later)
- **Coarse-grid approximations** - For a PDE discretized on a grid, a preconditioner can be formed by transferring the solution to a coarser grid, solving a smaller problem, then transferring back (*multigrid*)

Incomplete Cholesky Factorization (IC, ILU)

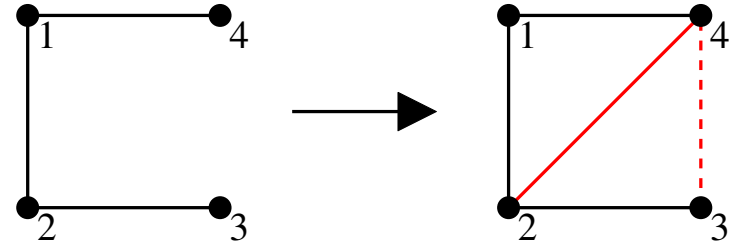
$$\begin{bmatrix} \bullet & & & \\ & \bullet & & \\ & & \bullet & \\ & & & \bullet \end{bmatrix} \longrightarrow \begin{bmatrix} \bullet & & & \\ & \bullet & & \\ & & \bullet & \\ & & & \bullet \end{bmatrix} \times \begin{bmatrix} \bullet & & & \\ & \bullet & & \\ & & \bullet & \\ & & & \bullet \end{bmatrix}$$

$A \qquad R^T \qquad R$

- Compute factors of A by Gaussian elimination, but ignore fill
- Preconditioner $B = R^T R \approx A$, not formed explicitly
- Compute $B^{-1}z$ by triangular solves in time $O(\text{nnz}(A))$
- Total storage is $O(\text{nnz}(A))$, static data structure
- Either symmetric (IC) or nonsymmetric (ILU)

Incomplete Cholesky and ILU: Variants

- Allow one or more “levels of fill”
 - Unpredictable storage requirements



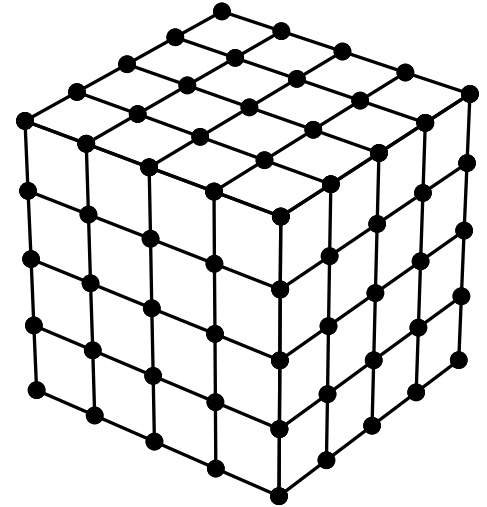
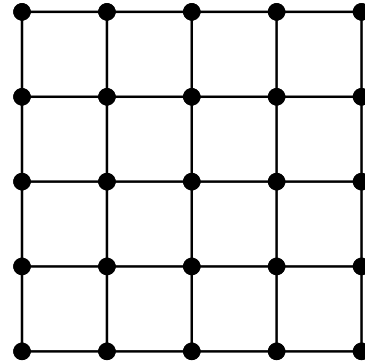
- Allow fill whose magnitude exceeds a “drop tolerance”
 - May get better approximate factors than levels of fill
 - Unpredictable storage requirements
 - Choice of tolerance is ad hoc
- Partial pivoting (for nonsymmetric A)
- “Modified ILU” (MIC): Add dropped fill to diagonal of U or R
 - A and $R^T R$ have same row sums
 - Good in some PDE contexts

Incomplete Cholesky and ILU: Issues

- Choice of parameters
 - Good: Smooth transition from iterative to direct methods
 - Bad: Very ad hoc, problem-dependent
 - Trade-off: Time per iteration vs # of iterations (more fill \rightarrow more time but fewer iterations)
- Effectiveness
 - Condition number usually improves (only) by constant factor (except MIC for some problems from PDEs)
 - Still, often good when tuned for a particular class of problems
- Parallelism
 - Triangular solves are not very parallel
 - Reordering for parallel triangular solve by graph coloring

Complexity of Linear Solvers

- Time to solve the Poisson model problem on regular mesh with N nodes:



Solver	1-D	2-D	3-D
Sparse Cholesky	$O(N)$	$O(N^{1.5})$	$O(N^2)$
CG, exact arith.	$O(N^2)$	$O(N^2)$	$O(N^2)$
CG, no precondition.	$O(N^2)$	$O(N^{1.5})$	$O(N^{1.33})$
CG, modified IC	$O(N^{1.5})$	$O(N^{1.25})$	$O(N^{1.17})$
Multigrid	$O(N)$	$O(N)$	$O(N)$