

AMS526: Numerical Analysis I (Numerical Linear Algebra for Computational and Data Sciences)

Lecture 20: Direct Methods for Sparse Linear Systems; Overview of Iterative Methods

Xiangmin Jiao

Stony Brook University

Outline

1 Direct Methods for Sparse Linear Systems (MC§11.1-11.2)

2 Overview of Iterative Methods for Sparse Linear Systems

Banded Linear Systems

- Cost of factorizing banded linear system depends on bandwidth
 - ▶ For SPD $n \times n$ matrix with semi-bandwidth s , total flop count of Cholesky factorization is about ns^2
 - ▶ For $n \times n$ matrix with lower bandwidth p and upper bandwidth q ,
 - ★ In $A = LU$ (LU without pivoting), total flop count is about $2npq$
 - ★ In $PA = LU$ (LU with partial pivoting), total flop count is about $2np(p + q)$
- Banded matrices have their own special storage formats (such as Compressed Diagonal Storage (CDS))

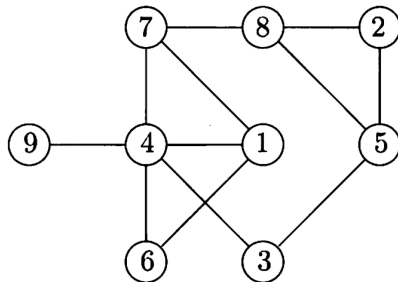
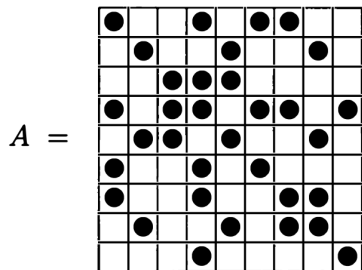
Fill

- When applying LU or Cholesky factorization to general sparse matrix, taking linear combinations of rows or columns to annihilate unwanted nonzero entries can introduce new nonzeros into matrix locations that were initially zero
- Such new nonzeros, called *fills* or *fill-ins*, must be stored and may themselves eventually need to be annihilated in order to obtain triangular factors
- Resulting triangular factors can be expected to contain at least as many nonzeros as original matrix and usually significant fills as well

Sparse Cholesky Factorization

- In general, some heuristic algorithms are employed to reorder the matrix to reduce fills
- Amount of fills is sensitive to order in which rows and columns of matrix are processed, so basic problem in sparse factorization is reordering matrix to limit fill during factorization
- Exact minimization of fills is hard combinatorial problem (NP-complete), but heuristic algorithms such as minimum degree and nested dissection limit fills well for many types of problems
- For Cholesky factorization, both rows and columns are reordered

Graph and Sparse Matrix



Graph Model of Elimination

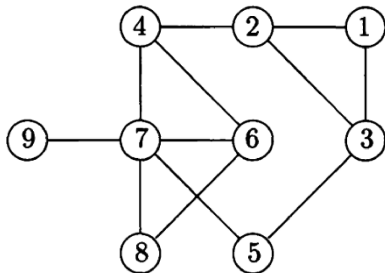
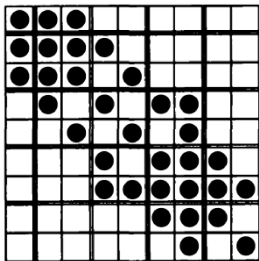
- Each step of factorization process corresponds to elimination of one node from graph
- Eliminating node causes its neighboring nodes to become connected to each other
- If any such neighbors were not already connected, then fill results (new edges in graph and new nonzeros in matrix)
- Commonly used reordering methods include Cuthill-McKee, approximate minimum degree ordering (AMD) and nested dissection

Reordering to Reduce Bandwidth

- The Cuthill-McKee algorithm and reverse Cuthill-McKee algorithm
 - ▶ The Cuthill-McKee algorithm is a variant of the breadth-first search algorithm on graphs.
 - ★ Starts with a peripheral node
 - ★ Generates levels R_i for $i = 1, 2, \dots$ until all nodes are exhausted
 - ★ The set R_{i+1} is created from set R_i by listing all vertices adjacent to all nodes in R_i
 - ★ Within each level, nodes are listed in ascending order by minimum predecessor (the already-visited neighbors) and as a tiebreak ascending by vertex degree
 - ▶ The reverse Cuthill-McKee algorithm (RCM) reserves the resulting index numbers

Example Cuthill-McKee Reordering

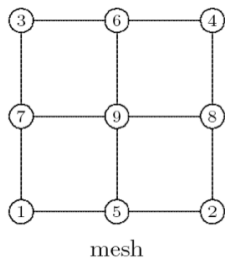
$$A(p,p) =$$



Approximate Minimum Degree Ordering

- Good heuristic for limiting fills is to eliminate first those nodes having fewest neighbors
- Number of neighbors is called *degree* of node, so heuristic is known as *minimum degree*
- At each step, select node of smallest degree for elimination, breaking ties arbitrarily
- After node has been eliminated, its neighbors become connected to each other, so degrees of some nodes may change
- Process is then repeated, with new node of minimum degree eliminated next, and so on until all nodes have been eliminated

Minimum Degree Ordering, continued



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

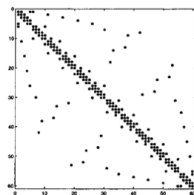
A

$$\begin{bmatrix}
 \times & & & & & & & & \\
 & \times & & & & & & & \\
 & & \times & & & & & & \\
 & & & \times & & & & & \\
 \times & \times & & & \times & & & & \\
 & & \times & \times & \times & & \times & & \\
 \times & & \times & & & + & + & \times & \\
 & \times & & \times & & + & + & + & \times \\
 & & & \times & \times & \times & \times & \times & \times
 \end{bmatrix}$$

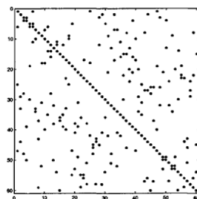
L

- Cholesky factor suffers much fewer fills than with original ordering, and advantage grows with problem size
- Sophisticated versions of minimum degree are among most effective general-purpose orderings known

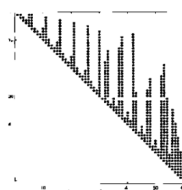
Comparison of Different Orderings of Example Matrix



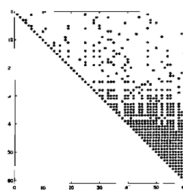
original ordering



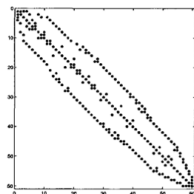
random reordering



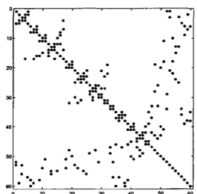
original ordering
(541 nonzeros)



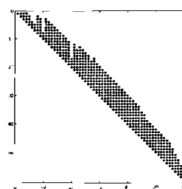
random reordering
(547 nonzeros)



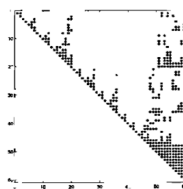
reverse Cuthill-McKee



minimum-degree ordering



reverse Cuthill-McKee
(514 nonzeros)



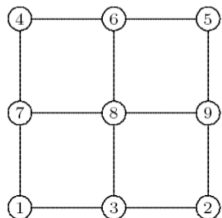
minimum-degree ordering
(360 nonzeros)

Left: Nonzero pattern of matrix A . Right: Nonzero pattern of matrix R .

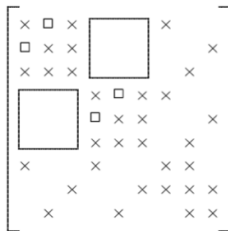
Nested Dissection Ordering

- Nested dissection is based on divide-and-conquer
- First, small set of nodes is selected whose removal splits graph into two pieces of roughly equal size
- No node in either piece is connected to any node in other, so no fill occurs in either piece due to elimination of any node in the other
- Separator nodes are numbered last, then process is repeated recursively on each remaining piece of graph until all nodes have been numbered

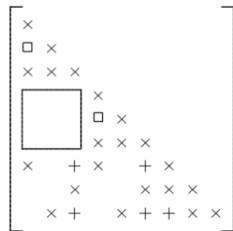
Nested Dissection Ordering Continued



mesh



A



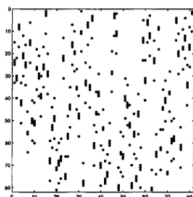
L

- Dissection induces blocks of zeros in matrix that are automatically preserved during factorization
- Recursive nature of algorithm can be seen in hierarchical block structure of matrix, which would involve many more levels in larger problems
- Again, Cholesky factor suffers much less fill than with original ordering, and advantage grows with problem size

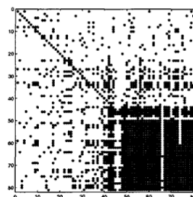
Sparse Gaussian Elimination

- For Gaussian elimination, only columns are reordered
- Pivoting introduces additional fills in sparse Gaussian elimination
- Reordering may be done dynamically or statically
- The reverse Cuthill-McKee algorithm applied to $A + A^T$ may be used to reduce bandwidth
- *Column approximate minimum-degree*, may be employed to reorder matrix to reduce fills

Comparison of Different Orderings of Example Matrix

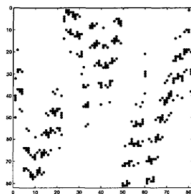


Spy plot of A

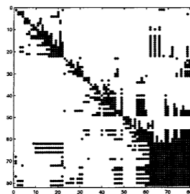


Spy plot of $L + U$
(2071 nonzeros)

Nonzero pattern of A and $L + U$ with random ordering.



Spy plot of A



Spy plot of $L + U$
(1249 nonzeros)

Nonzero pattern of A and $L + U$ with column AMD ordering.

Comparison of Direct Methods

- Computational cost for Laplace equation on $k \times k(\times k)$ grid with n unknowns

method	2-D		3-D	
dense Cholesky	k^6	n^3	k^9	n^3
banded Cholesky	k^4	n^2	k^7	$n^{2.33}$
sparse Cholesky	k^3	$n^{1.5}$	k^6	n^2

Reference: Michael T. Heath, *Scientific Computing: An Introductory Survey*, 2nd Edition, McGraw-Hill, 2002.

Software of Sparse Solvers

- Additional implementation complexities include cache performance and parallelism
- It is advisable to use software packages
- MATLAB has its own sparse solvers if matrix is stored in sparse format
 - ▶ Sparse matrix is created by using the “sparse” function
 - ▶ Reordering is implemented as “symrcm”, “symamd”, and “colamd”
- For symmetric matrices, an example is Taucs
- For non-symmetric matrices, popular tools include SuperLU, MUMPS, and PARDISO

Outline

- 1 Direct Methods for Sparse Linear Systems (MC§11.1-11.2)
- 2 Overview of Iterative Methods for Sparse Linear Systems

Direct vs. Iterative Methods

- *Direct methods*, or *noniterative methods*, compute the exact solution after a finite number of steps (in exact arithmetic)
 - ▶ Example: Gaussian elimination, QR factorization
- *Iterative methods* produce a sequence of approximations $x^{(1)}, x^{(2)}, \dots$ that hopefully converge to the true solution
 - ▶ Example: Jacobi, Conjugate Gradient (CG), GMRES, BiCG, etc.
- Caution: The boundary between direct and iterative methods is vague sometimes
- Why use iterative methods (instead of direct methods)?
 - ▶ may be faster than direct methods
 - ▶ produce useful intermediate results
 - ▶ handle sparse matrices more easily (needs only matrix-vector product)
 - ▶ often are easier to implement on parallel computers
- Question: When not to use iterative methods?

Two Classes of Iterative Methods

- Stationary iterative methods are fixed-point iterations obtained by matrix splitting
 - ▶ Examples: Jacobi (for linear systems, not Jacobi iterations for eigenvalues), Gauss-Seidel, Successive Over-Relaxation (SOR) etc.
- *Krylov subspace methods* find optimal solution in *Krylov subspace* $\{b, Ab, A^2b, \dots, A^k b\}$
 - ▶ Build subspace successively
 - ▶ Example: Conjugate Gradient (CG), Generalized Minimum Residual (GMRES), BiCG, etc.
 - ▶ We will focus on Krylov subspace methods

Stationary Iterative Methods

- *Stationary iterative methods* find a splitting $A = M - N$ and iterates

$$x_{k+1} = M^{-1}(Nx_k + b)$$

- Suppose $r_k = b - Ax_k$, we have $x_* = x_k + A^{-1}r_k$.
- Stationary iterative method approximates it by

$$x_{k+1} = x_k + M^{-1}r_k$$

because

$$\begin{aligned}x_{k+1} &= M^{-1}Nx_k + M^{-1}b \\&= M^{-1}Nx_k + M^{-1}(r_k + Ax_k) \\&= M^{-1}(N + A)x_k + M^{-1}r_k \\&= x_k + M^{-1}r_k\end{aligned}$$

- A stationary iterative method is good if
 - ▶ $\rho(M^{-1}N) < 1$, and
 - ▶ M^{-1} is a good approximation to A^{-1}

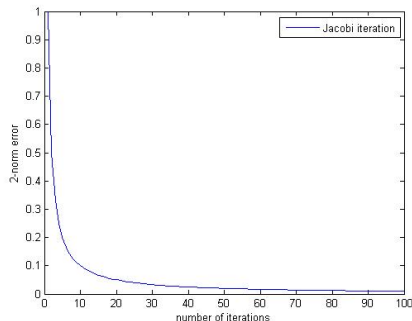
Stationary Iterative Methods

- Different choices of splitting will lead to various schemes
- Let $A = L + D + U$, where D is diagonal, L is strictly lower triangular, and U is strictly upper triangular
 - ▶ Jacobi iteration: $M = D$, works well if A is diagonally dominant
 - ▶ Gauss-Seidel: $M = L + D$, works well if A is SPD
 - ▶ Successive Over-Relaxation (SOR): $M = \frac{1}{\omega}D + L$, where $1 \leq \omega < 2$, converges quickly proper choice of ω
 - ▶ Symmetric SOR: symmetric version of SOR
- These methods work for some problems, but they may converge slowly
- Nevertheless, stationary methods are important as preconditioners for Krylov-subspace methods and smoothers in multigrid methods (later)

Stationary Iterative Methods

Example

For 2D Poisson equation, spectral radius of Jacobi iteration matrix is $\cos\left(\frac{\pi}{n}\right) \approx 1 - O\left(\frac{1}{n^2}\right)$. Number of iterations required to achieve ϵ is $O(n^2 \ln \epsilon^{-1})$.



After 5 Jacobi iterations on a Poisson equation, error decreases very slowly.