Scientific Computing with Case Studies
SIAM Press, 2009
http://www.cs.umd.edu/users/oleary/SCCSwebpage
Lecture Notes for Unit VII
Sparse Matrix Computations
Part 1: Direct Methods
Dianne P. O'Leary

Solving Sparse Linear Systems

Assumed background from Chapter 5:

- Gauss elimination and the LU decomposition
- (Partial) Pivoting for stability
- Cholesky decomposition of a symmetric positive definite matrix
- Forward and backward substitution for solving lower or upper triangular systems

---

The plan:

- How to store a sparse matrix
- Sparse direct methods
  - The difficulty with fill-in
  - Some strategies to reduce fill-in
- Iterative methods for solving linear systems:
  - Basic (slow) iterations: Jacobi, Gauss-Seidel, SOR.
  - Krylov subspace methods
  - Preconditioning (where direct meets iterative)
- A special purpose method: Multigrid

Note: There are sparse direct and Krylov subspace methods for eigenvalue problems, but we will just mention them in passing.

Contrasting direct and iterative methods

- Direct: Usually the method of choice for 2-d PDE problems.
- Iterative: Usually the method of choice for 3-d PDE problems.
- Direct: Except for round-off errors, we produce an exact solution to our problem.
  Iterative: We produce an approximate solution with a given tolerance.

- Direct: We usually require more storage (sometimes much more) than the original
  Iterative: We only require a few extra vectors of storage.
- Direct: The elements of the matrix **A** are modified by the algorithm.
  Iterative: We don't need to store the elements explicitly; we only need a function that can form **Ax** for any given vector **x**.
- Direct: If a new **b** is given to us later, solving the new system is fast.
  Iterative: It is not as easy to solve the new system.
- Direct: The technology is well developed and good software is standard.
  Iterative: Some software exists, but it is incomplete and rapidly evolving.

Reference for direct methods: Chapter 27.
Reference for iterative methods: Chapter 28.

Storing a sparse matrix

There are many possible schemes. MATLAB chooses a typical one: store the indices and values of the nonzero elements in column order, so

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 5 & 7 & 0 \\ 1 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \quad \text{is stored as} \quad \begin{matrix} (1,1) & 2 \\ (3,1) & 1 \\ (2,2) & 5 \\ (2,3) & 7 \\ (3,3) & 6 \\ (4,4) & 8 \end{matrix}$$

Therefore, storing a sparse matrix takes about $3nz$ storage locations, where nz is the number of nonzeros.

Sparse direct methods

Recommended references:

- In MATLAB, type `demo`, and then choose "Matrices", "Sparse Matrices".
- Also In MATLAB, type `demo`, and then choose "Matrices", "Orderings and Separators for a Finite Element Matrix" (but don't worry about the trees that it produces).

The problem:

We want to solve

$$\mathbf{Ax} = \mathbf{b}.$$

For simplicity, we'll assume for now that

- **A** symmetric.

- **A** positive definite.

For a PDE, this is ok if the underlying operator $\mathcal{A}$ is self-adjoint and coercive.

The difficulty with fill-in: A motivating example

Suppose we want to solve a system involving an $n \times n$ arrowhead matrix:

$$\mathbf{A} = \begin{bmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ \times & 0 & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & 0 & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix}$$

where $\times$ denotes a nonzero value (we don't care what it is) and 0 denotes a zero. The number of nonzeros is $3n - 2$.

Suppose we use Gauss-elimination (or the LU factorization, or the Cholesky factorization – all of them have the same trouble).

Then in the first step, we add some multiple of the first row to every other row. Disaster! The matrix is now fully dense with n^2 nonzeros!

A fix

Let's rewrite our problem by moving the first column and the first row to the end:

$$\mathbf{A} = \begin{bmatrix} \times & 0 & 0 & 0 & 0 & \times \\ 0 & \times & 0 & 0 & 0 & \times \\ 0 & 0 & \times & 0 & 0 & \times \\ 0 & 0 & 0 & \times & 0 & \times \\ 0 & 0 & 0 & 0 & \times & \times \\ \times & \times & \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_1 \end{bmatrix} = \begin{bmatrix} b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_1 \end{bmatrix}$$

Now when we use Gauss-elimination (or the LU factorization, or the Cholesky factorization) no new nonzeros are produced in A.

Reordering the variables and equations is a powerful tool for maintaining sparsity during factorization.

Strategies for overcoming fill-in

We reorder the rows and columns with a permutation matrix **P** and solve

$$\mathbf{PAP}^T(\mathbf{Px}) = \mathbf{Pb}$$

instead of $\mathbf{Ax} = \mathbf{b}$.

Note that $\tilde{\mathbf{A}} = \mathbf{P}\mathbf{A}\mathbf{P}^T$ is still symmetric and positive definite, so we can use the Cholesky factorization $\tilde{\mathbf{A}} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ where $\mathbf{L}$ is lower triangular with ones on its diagonal and $\mathbf{D}$ is diagonal.

How to reorder

- Finding the optimal reordering is generally too expensive: it is in general an NP hard problem.
- Therefore, we rely on heuristics that give us an inexpensive algorithm to find a reordering.
- As a consequence, usually the heuristics do well, but sometimes they produce a very bad reordering.

Important insights

Insight 1: If $\mathbf{A}$ is a band matrix, i.e.,

- tridiagonal,
- pentadiagonal,
- ...,

then there is never any fill outside the band.

Insight 2: Also, there is never any fill outside the profile of a matrix, where the profile stretches from the first nonzero in each column to the main diagonal element in the column, and from the first nonzero in each row to the main diagonal element.

Example: Zeros within the profile marked with $\otimes$

$$\mathbf{A} = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & \times \end{bmatrix}, \quad \text{profile}(\mathbf{A}) = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & \otimes & 0 & \times & 0 \\ 0 & 0 & \times & 0 & \times & 0 \\ \times & \otimes & \otimes & \times & \otimes & 0 \\ 0 & 0 & \times & \otimes & \times & 0 \\ 0 & \times & \otimes & \otimes & \otimes & \times \end{bmatrix},$$

So some methods try to produce a reordered matrix with a small band or a small profile.

Insight 3: The sparsity of a matrix can be encoded in a graph. For example, a symmetric matrix

$$\mathbf{A} = \begin{bmatrix} \times & 0 & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & \times \\ \times & 0 & \times & 0 & \times & 0 \\ 0 & 0 & 0 & \times & 0 & 0 \\ 0 & \times & \times & 0 & \times & 0 \\ 0 & \times & 0 & 0 & 0 & \times \end{bmatrix}$$

has upper-triangular nonzero off-diagonal elements $a_{13}, a_{25}, a_{26}, a_{35}$ and corresponds to a graph with 6 nodes, one per row/column, and edges connecting nodes $(1,3)$, $(2,5)$, $(2,6)$, and $(3,5)$.

Unquiz: Draw the graph corresponding to the finite difference matrix for Laplace's equation on a 5 x 5 grid. []

Some reordering strategies

Strategy 1: Cuthill-McKee

One of the oldest is Cuthill-McKee, which uses the graph to order the rows and columns:

> Find a starting node with minimum degree (degree = number of neighbors). Until all nodes are ordered,
>
> > For each node that was ordered in the previous step, order all of the unordered nodes that are connected to it, in order of their degree.

Reverse Cuthill-McKee (doing a final reordering from last to first) often works even better.

Strategy 2: Minimum Degree

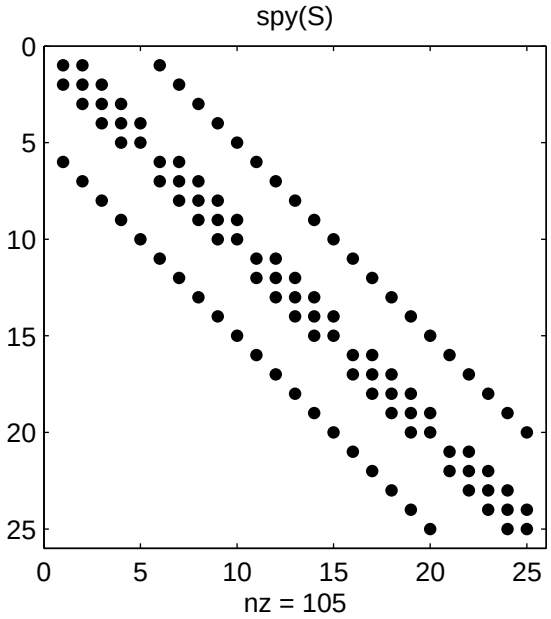
Until all nodes are ordered,

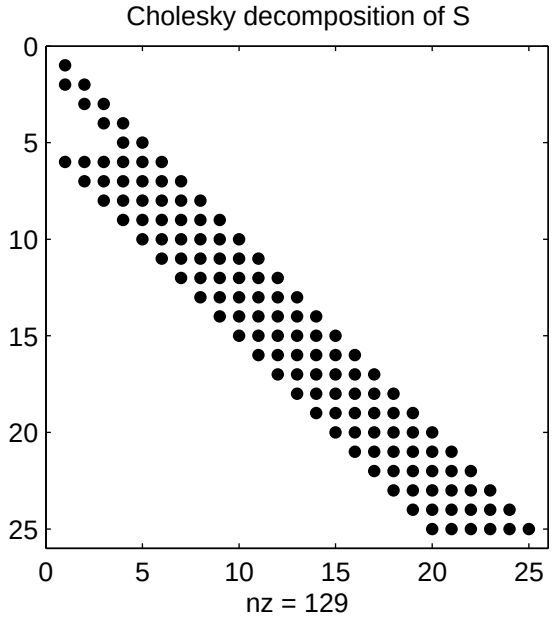
> Choose a node that has the smallest degree, and order that node next, removing it from the graph. (If there is a tie, choose any of the candidates.)

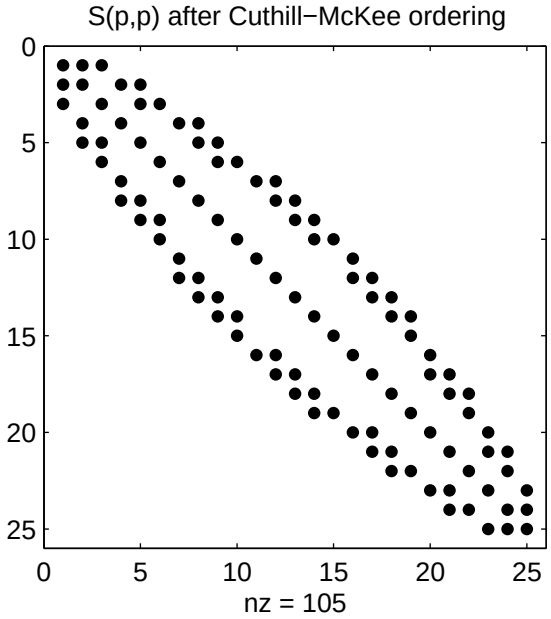
Strategy 3: Nested Dissection

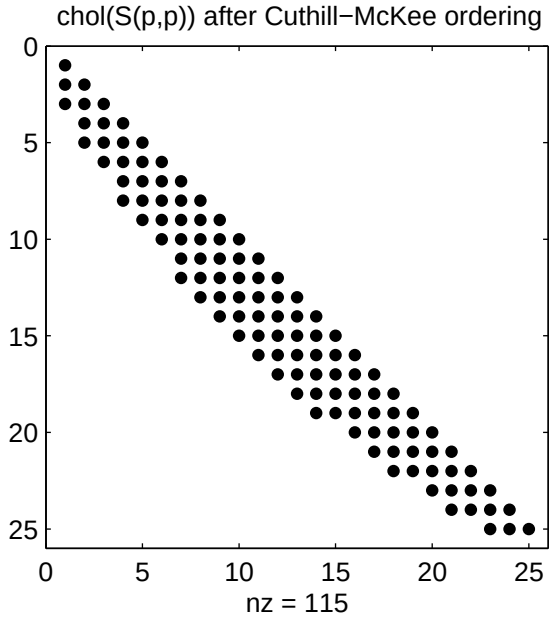
Try to break the graph into 2 pieces plus a separator, with

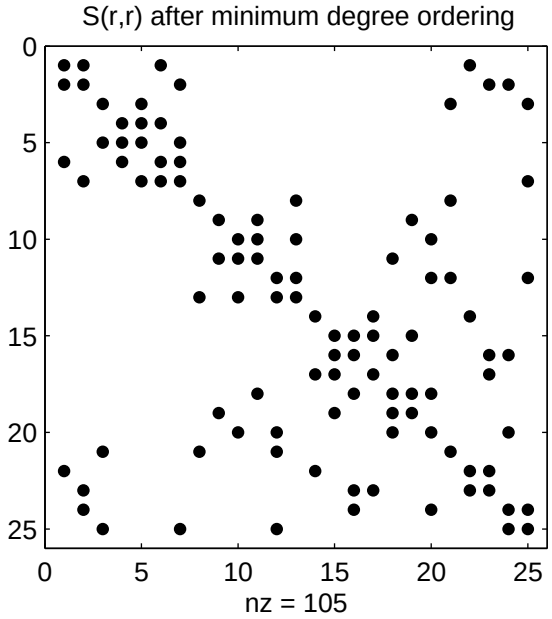
- approximately the same number of nodes in the two pieces,

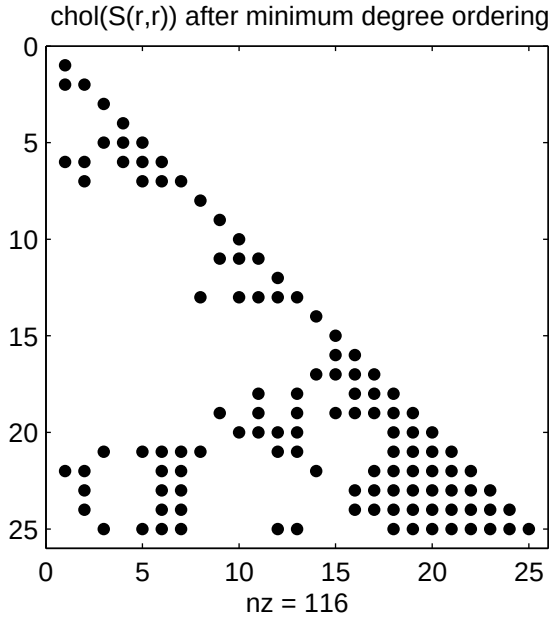

spy(S)
0
5
10
15
20
25
0
5
10
15
20
25
nz = 105


Cholesky decomposition of S
0
5
10
15
20
25
0
5
10
15
20
25
nz = 129


S(p,p) after Cuthill−McKee ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 105


chol(S(p,p)) after Cuthill−McKee ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 115


S(r,r) after minimum degree ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 105


chol(S(r,r)) after minimum degree ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 116

- no edges between the two pieces,
- a small number of nodes in the separator.

Do this recursively until all pieces have a small number of nodes.

Then order the nodes piece by piece. Finally, order the nodes in the separators.

A comparison of the orderings

Demo: spar50.m

Summary of sparse direct methods

- The idea is to reorder rows and columns of the matrix in order to reduce fill-in during the factorization.
- We have given examples of strategies useful for symmetric positive definite matrices.
- For nonsymmetric matrices, and for symmetric indefinite matrices, there is an additional critical complication: we need to reorder for stability as well as sparsity preservation.
- For nonsymmetric matrices, strategies are similar, but the row permutation is allowed to be different from the column permutation, since there is no symmetry to preserve.
- The most recent reordering strategies are based on partitioning the matrix by spectral partitioning, using the elements of an eigenvector of a matrix related to the graph. These methods are becoming more popular.

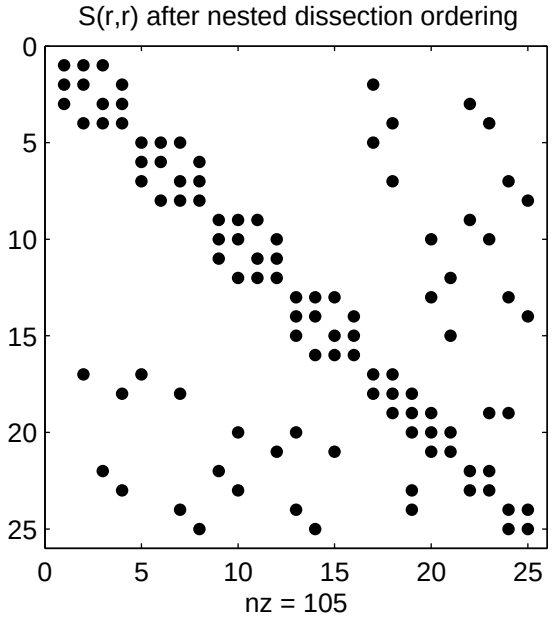
Sparse direct methods for eigenproblems

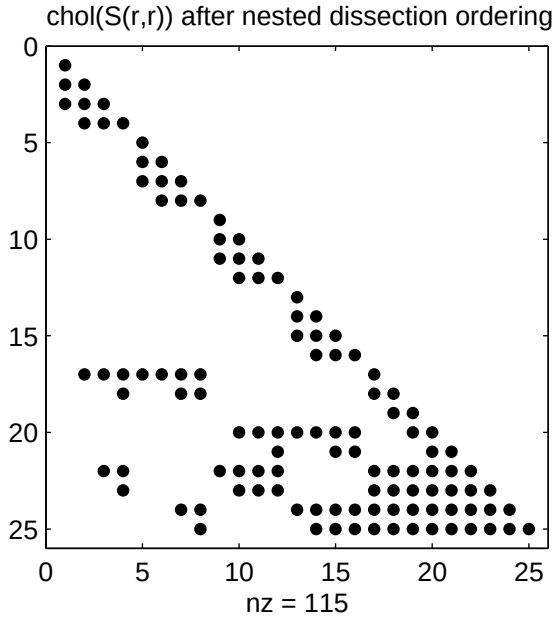
The standard algorithm for finding eigenvalues and eigenvectors: the QR algorithm.

In MATLAB, `[V,D] = eig(A)` puts the eigenvectors of $\mathbf{A}$ in the columns of $\mathbf{V}$ and the eigenvalues along the main diagonal of $\mathbf{D}$.

Strategy for sparse matrices:

- Use a reordering strategy to reduce the bandwidth of the matrix. We replace $\mathbf{A}$ by $\mathbf{PAP}^T$, which has the same eigenvalues but eigenvectors equal to $\mathbf{P}$ times the eigenvectors of $\mathbf{A}$.
- Use the QR algorithm to find the eigenvalues of the resulting banded matrix.


S(r,r) after nested dissection ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 105


chol(S(r,r)) after nested dissection ordering
0
5
10
15
20
25
0
5
10
15
20
25
nz = 115

Actually, MATLAB's `eig` will not even attempt to find the eigenvectors of a matrix stored in sparse format, because the matrix $\mathbf{V}$ is generally full, but it will compute the eigenvalues.

If we want (a few) eigenvalues and eigenvectors of a sparse matrix, we use MATLAB's `eigs` function, which uses a Krylov subspace method based on the Arnoldi basis, to be discussed later.