Combinatorial Scientific Computing

"I observed that most of the coefficients in our matrices were zero; i.e., the nonzeros were 'sparse' in the matrix, and that typically the triangular matrices associated with the forward and back solution provided by Gaussian elimination would remain sparse if pivot elements were chosen with care"

- Harry Markowitz, describing the 1950s work on portfolio theory that won the 1990 Nobel Prize for Economics

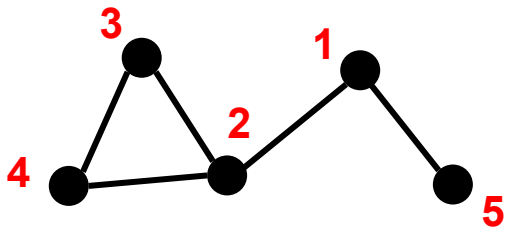
UCSB

Sparse Cholesky factorization to solve Ax = b

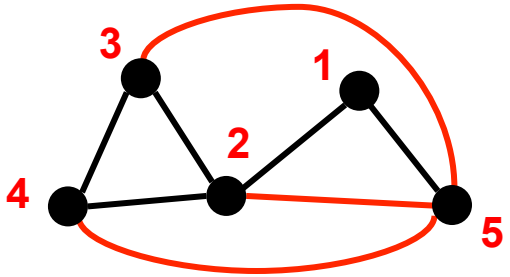
1. Preorder: replace A by PAP^T and b by Pb
   - Independent of numerics
2. Symbolic Factorization: build static data structure
   - Elimination tree
   - Nonzero counts
   - Supernodes
   - Nonzero structure of L
3. Numeric Factorization: $A = LL^T$
   - Static data structure
   - Supernodes use BLAS3 to reduce memory traffic
4. Triangular Solves: solve $Ly = b$, then $L^Tx = y$

Sparse Gaussian elimination and chordal completion

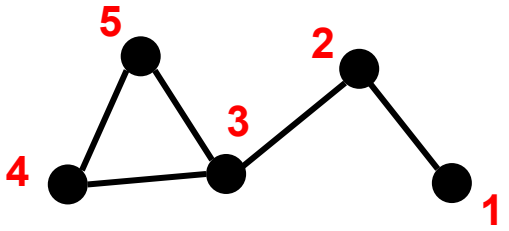
[Parter, Rose]



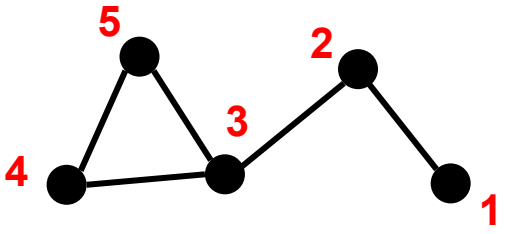
$$Ax = b$$



$$A = L_1 L_1^T$$



$$(PAP^T)\,(Px) = (Pb)$$

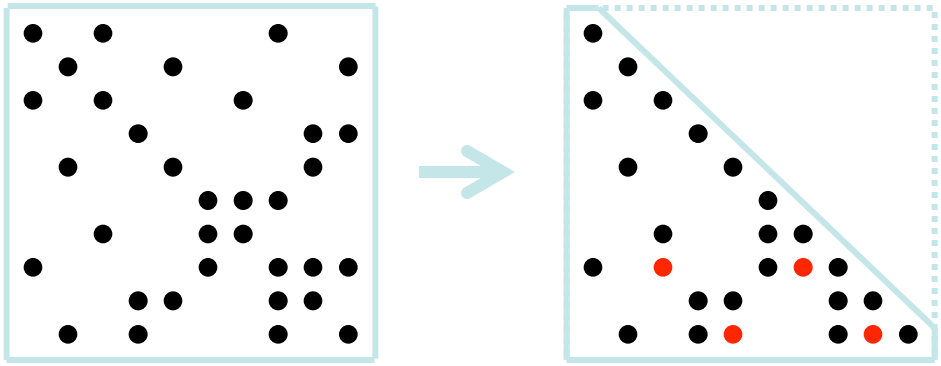


$$PAP^T = L_2 L_2^T$$

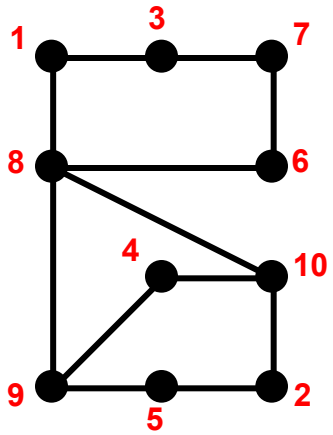
UCSB

Graphs and sparse matrices: Cholesky factorization

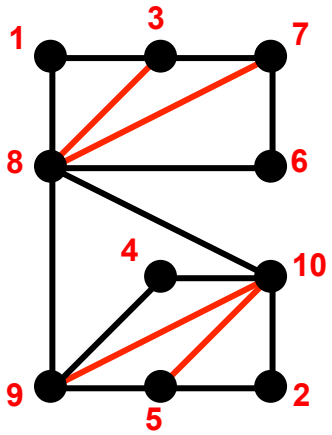
$$A = LL^T$$



Fill: new nonzeros in factor



G(A)



G(L)
[chordal]

Symmetric Gaussian elimination:

for j = 1 to n
    add edges between j's higher-numbered neighbors

UCSB

Sparse Gaussian elimination and chordal completion

[Parter, Rose]

Repeat:

Choose a vertex v and mark it;

Add edges between unmarked neighbors of v;

Until: Every vertex is marked

Goal: End up with as few edges as possible.

Or, add fewest possible edges to make the graph chordal.

Space = edges + vertices = $\sum_{\text{vertices}}$ (1 + # higher neighbors)

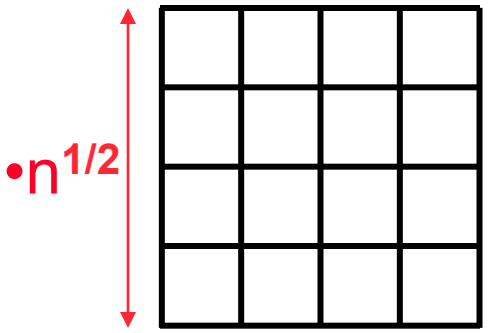
Time = flops = $\sum_{\text{vertices}}$ (1+ # higher neighbors)^2

UCSB

Complexity measures for sparse Cholesky

- Space:
  - Measured by fill, which is nnz($G^+(A)$)
  - Number of off-diagonal nonzeros in Cholesky factor
    (need to store about n + nnz($G^+(A)$) real numbers).
  - Sum over vertices of $G^+(A)$ of (# of higher neighbors).
- Time:
  - Measured by number of flops (multiplications, say)
  - Sum over vertices of $G^+(A)$ of (# of higher neighbors)^2
- Front size:
  - Related to the amount of "fast memory" required
  - Max over vertices of $G^+(A)$ of (# of higher neighbors).

The (2-dimensional) model problem



- Graph is a regular square grid with $n = k^2$ vertices.
- Corresponds to matrix for regular 2D finite difference mesh.
- Gives good intuition for behavior of sparse matrix algorithms on many 2-dimensional physical problems.
- There's also a 3-dimensional model problem.

Permutations of the 2-D model problem

- Theorem 1: With the natural permutation, the n-vertex model problem has exactly $O(n^{3/2})$ fill.

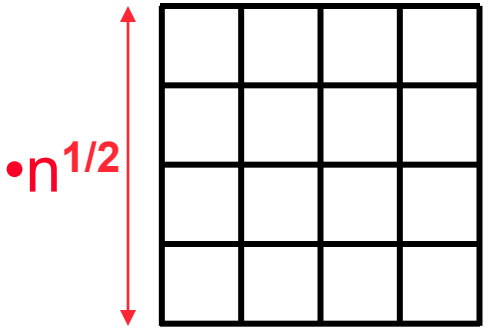
- Theorem 2: With a *nested dissection* permutation, the n-vertex model problem has exactly $O(n \log n)$ fill.

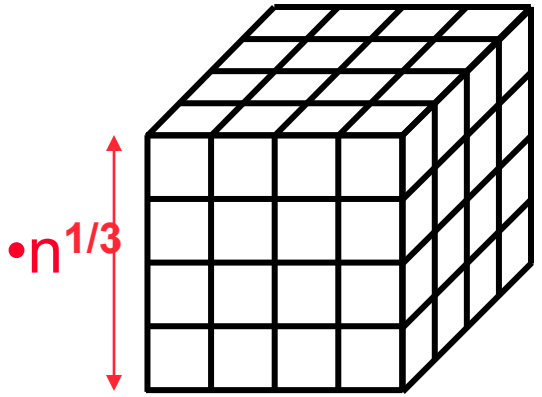
- Theorem 3: With any permutation, the n-vertex model problem has at least $O(n \log n)$ fill.

See course notes for proofs.

Complexity of direct methods

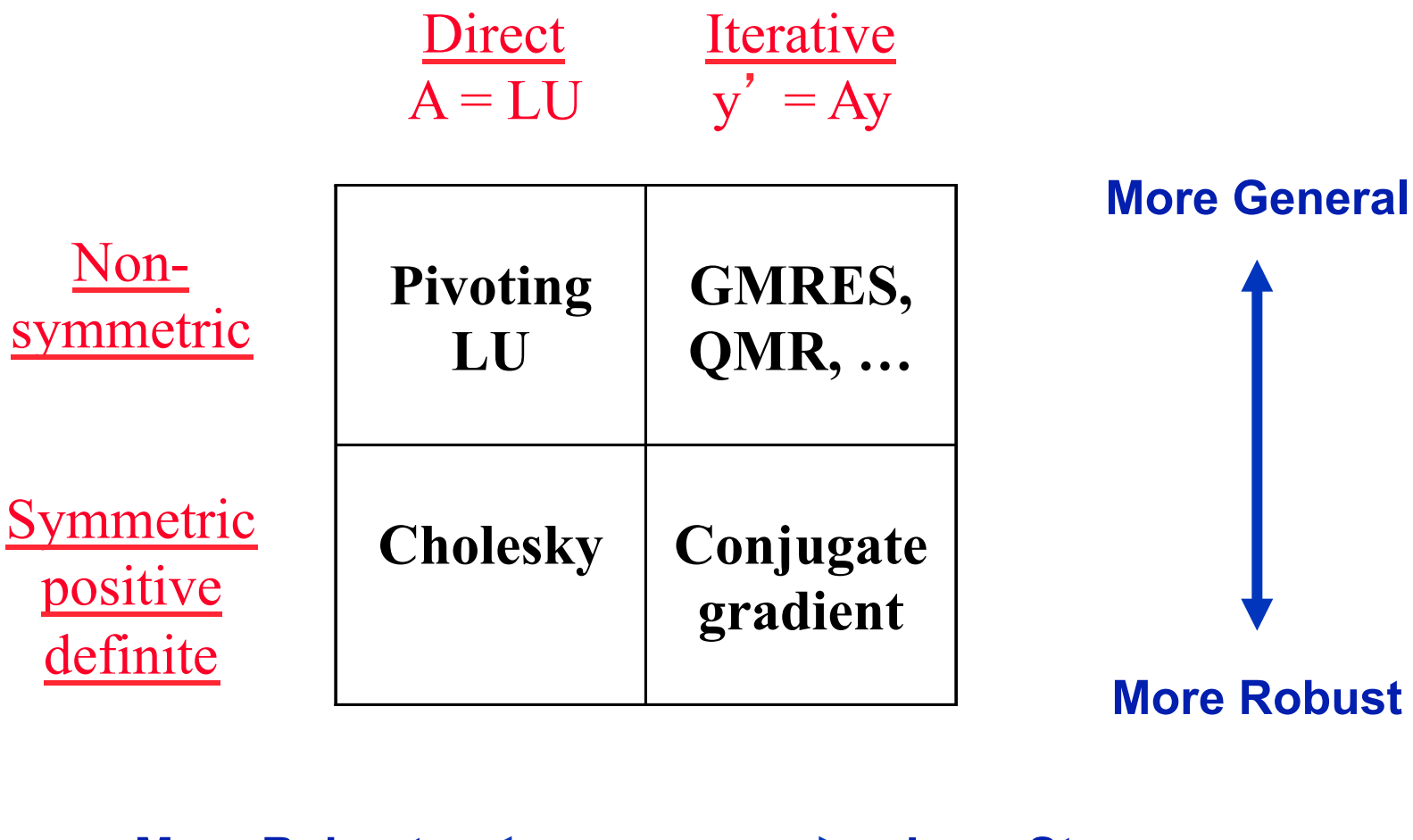
- Time and space to solve any problem on any well-shaped finite element mesh





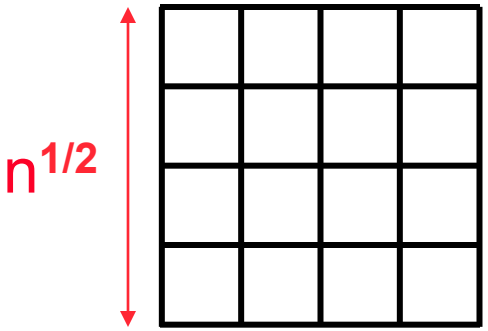
	2D	3D
Space (fill):	$O(n \log n)$	$O(n^{4/3})$
Time (flops):	$O(n^{3/2})$	$O(n^{2})$

The Landscape of Sparse Ax=b Solvers

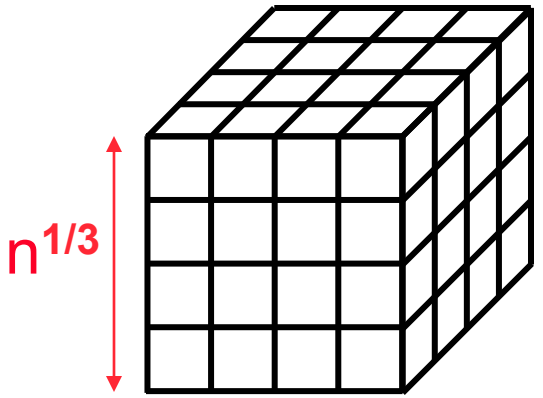


Complexity of linear solvers

Time to solve model problem (Poisson's equation) on regular mesh



2D



3D

	2D	3D
Dense Cholesky:	$O(n^3)$	$O(n^3)$
Sparse Cholesky:	$O(n^{1.5})$	$O(n^2)$
CG, exact arithmetic:	$O(n^2)$	$O(n^2)$
CG, no precond:	$O(n^{1.5})$	$O(n^{1.33})$
CG, modified IC:	$O(n^{1.25})$	$O(n^{1.17})$
CG, support trees:	$O(n^{1+})$	$O(n^{1+})$
Multigrid:	$O(n)$	$O(n)$