Structure prediction, elimination process on the graphs and fill-reducing ordering methods

Bora Uçar

RO:MA, LIP, ENS Lyon, France

CR-08: Combinatorial scientific computing, October 2014
http://perso.ens-lyon.fr/bora.ucar/CR08/
Outline

1. Sparse triangular system solutions
2. Predicting the structure of factorization
   - Elimination process on the graphs
3. Ordering sparse matrices
   - Fill-reducing orderings
   - Local fill-reducing ordering methods
   - Minimum fill-in heuristic
   - Global fill-reducing ordering methods
4. Closing
Outline

1. Sparse triangular system solutions
2. Predicting the structure of factorization
   - Elimination process on the graphs
3. Ordering sparse matrices
   - Fill-reducing orderings
   - Local fill-reducing ordering methods
   - Minimum fill-in heuristic
   - Global fill-reducing ordering methods
4. Closing
Solve $Lx = b$

We are going to solve $Lx = b$ with forward substitution. At index $i$, we need to solve

$$\sum_{j<i} \ell_{ij} x_j + \ell_{ii} x_i = b_i.$$ 

When do we subtract $\sum_{j<i} \ell_{ij} x_j$ from $b_i$?
Solve $Lx = b$

**Row oriented**

\[
\begin{align*}
\text{for } i = 1 \text{ to } n \text{ do} & \\
& x(i) \leftarrow b(i) \\
& \text{for } j = 1 \text{ to } i - 1 \text{ do} \\
& \quad x(i) \leftarrow x(i) - L(i, j) \times x(j) \\
& x(i) \leftarrow x(i)/L(i, i)
\end{align*}
\]

Pull based: to compute $x_i$, pull other, known $x$.
$L$ is accessed by rows.

**Column oriented**

\[
\begin{align*}
& x(1 : n) \leftarrow b(1 : n) \\
\text{for } j = 1 \text{ to } n \text{ do} & \\
& \quad x(j) \leftarrow x(j)/L(j, j) \\
& \quad x(j + 1 : n) \leftarrow x(j + 1 : n) - \\
& \quad \quad \quad L(j + 1 : n, j) \times x(j)
\end{align*}
\]

Push based: when $x_i$ is computed, inform others.
$L$ is accessed by columns.

- Both versions run in $O(nnz(L))$ time.
- For dense $b$ and $x$, it is ok.
- How one can exploit sparsity of $b$ and $x$?

(some parts of the slides are from John Gilbert)
Sparse triangular matrices and dags

- The directed graph $G(A)$ of a matrix $A$ is such that $a_{i,j} \neq 0$ iff $(i,j)$ is an edge in $G(A)$.
- The directed graph of a triangular matrix is directed acyclic.
The structure of the solution vector $x$ of $Lx = b$ is given by the set of vertices reachable from vertices of $b$ by paths in the dag of $G(L^T)$.

**Symbolic:** predict the structure of $x$ with depth-first-search from nonzeros of $b$.

**Numeric:** compute the values of $x$ in topological order.
Column oriented algorithm

DFS in $G(L^T)$ to predict nonzeros of $x$

$x(1: n) \leftarrow b(1: n)$

for $j =$ nonzero indices of $x$ in topological order do
  $x(j) \leftarrow x(j)/L(j, j)$
  for each $i$ in $L(j + 1: n, j)$ do
    $x(i) \leftarrow x(i) - L(i, j) \times x(j)$
DFS visits only the necessary edges and vertices. Not $O(n + \tau)$. In other words not $O(|V| + |E|)$.

Accesses only to the necessary components of $x$ (and of course that of $L$ as before) and performs only necessary operations (no zeros).

Therefore, a lower (or an upper) triangular system $Lx = b$ can be solved in $O(flops(Lx))$ time, if $L$ is stored by columns.

Is there a similar algorithm if $L$ is stored by rows? (J. Gilbert)

No $O(L)$ additional storage is allowed. Where is the difficulty?

Can assume reasonable things: that the graph of $L + L^T$ is chordal, i.e., for $i < j < k$, if $\ell_{ji}$ and $\ell_{ki}$ are nonzero, so is $\ell_{kj}$; or an $O(n)$ initialization for $n$ solves that will follow.
Outline

1. Sparse triangular system solutions

2. Predicting the structure of factorization
   - Elimination process on the graphs

3. Ordering sparse matrices
   - Fill-reducing orderings
   - Local fill-reducing ordering methods
   - Minimum fill-in heuristic
   - Global fill-reducing ordering methods

4. Closing
Structure prediction

From: Gilbert,'94 [Predicting structure]

\( f \) be a function from one or more matrices/vectors to a matrix/vector. We want to determine the structure of \( f(A) \) using \( A \).

\( A \) is not always enough: sum of two vectors is full, but \( (1, 1)^T + (1, -1)^T \) is not.

We ignore zeros created by coincidence in the numerical values of \( A \) and determine the smallest structure that is big enough for the result of \( f \) with any given input of the given structure. Given \( f \) and \( \text{struct}(A) \), determine

\[
\bigcup_{B} \{ \text{struct}(f(B)) : \text{struct}(B) \subseteq \text{struct}(A) \}
\]
Predicting structure helps
- in reducing the memory requirements,
- in achieving high performance,
- in simplifying the algorithms.

We will consider the Cholesky factorization \( A = LL^T \). In this case, structural and numerical aspects are neatly separated. We will also touch \( A = LU \) without pivoting.

For general case (e.g., LU factorization) pivoting is necessary and depends on the actual numerical values. There are specialized combinatorial tools for these. The methods that are used for \( LL^T \) are viable in this case too, hence our focus.

Structure prediction algorithms should run, preferably, faster than the numerical computations that will follow.
Assumptions: \( A \) symmetric (positive definite; diagonals are \( > 0 \) even during elimination \( > 0 \))

Structure of \( A \) (symmetric) is represented by the graph \( G = (V, E) \)

- Vertices are associated to columns: \( V = \{1, \ldots, n\} \)
- Edges \( E \) are defined by: \( (i, j) \in E \iff a_{ij} \neq 0 \)
- \( G \) undirected (symmetry of \( A \))
Symmetric matrices and graphs

- Number of nonzeros in column $j$ is $d(j) = |\text{adj}_G(j)|$
Let $A$ be a symmetric positive definite matrix of order $n$

The $LL^T$ factorization can be described by the equation:

$$A = A_0 = H_0 = \begin{pmatrix} d_1 & v_1^T \\ v_1 & H_1 \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{d_1} & 0 \\ \frac{v_1}{\sqrt{d_1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix} \begin{pmatrix} \sqrt{d_1} & v_1^T \\ 0 & d_1 \end{pmatrix}$$

$$= L_1 A_1 L_1^T,$$ 

where

$$H_1 = \bar{H}_1 - \frac{v_1 v_1^T}{d_1}$$

The basic step is applied on $H_1 H_2 \cdots$ to obtain:

$$A = (L_1 L_2 \cdots L_{n-1}) I_n \left( L_{n-1}^T \cdots L_2^T L_1^T \right) = LL^T$$
The basic step: $H_1 = \overline{H_1} - \frac{v_1v_1^T}{d_1}$

What is $v_1v_1^T$ in terms of structure?

$v_1$ is a column of $A$, hence the neighbors of the corresponding vertex.

$v_1v_1^T$ results in a dense sub-block in $H_1$.

If any of the nonzeros in dense submatrix are not in $A$, then we have fill-ins.
The elimination process in the graphs

\[ G_U(V, E) \leftarrow \text{undirected graph of } A \]

\begin{verbatim}
for k = 1 : n - 1 do
    V ← V - {k} ▶ remove vertex k
    E ← E - \{(k, ℓ) : ℓ ∈ adj(k)\} ∪ \{(x, y) : x ∈ adj(k) and y ∈ adj(k)\}
    G_k ← (V, E) ▶ for definition
\end{verbatim}

\( G_k \) are the so-called elimination graphs (Parter,'61).

![Diagram showing the elimination process on graphs](image)
A sequence of elimination graphs
Fill-in and operation count

In the light of the elimination process,

what is the total number of nonzeros in $L$?

what is the operation count incurred while forming $L$?
In the light of the elimination process,
what is the total number of nonzeros in \( L \)?
what is the operation count incurred while forming \( L \)?
express them using the degree of the vertices: \( \sum d(i) \) and \( \sum d(i)^2 \)
Elimination process: Formal definitions

For an undirected graph $G = (V, E)$ with $|V| = n$, an ordering of $V$ is a bijection $\alpha : \{1, \ldots, n\} \leftrightarrow V$. $\alpha^{-1}(v)$ gives the order of $v$.

$G_\alpha = (V, E, \alpha)$ is an ordered graph.

Deficiency of a vertex: $D(v)$ is the set of edges defined by $D(v) = \{(x, y) : x \in \text{adj}(v) \text{ and } y \in \text{adj}(v) \text{ and } y \notin \text{adj}(x) \text{ and } x \neq y\}$

$v$-elimination graph: Apply the elimination process to the vertex $v$ of $G$ to obtain $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v))$.

For an ordered graph $G_\alpha = (V, E, \alpha)$, the elimination process $P(G_\alpha) = [G = G_0, G_1, G_2, \ldots, G_{n-1}]$ is the sequence of elimination graphs defined by $G_0 = G, G_i = (G_{i-1})_{\alpha(i)}$. 
Elimination process: Formal definitions

For a directed graph $G = (V, E)$ with $|V| = n$, an ordering of $V$ is a bijection $\alpha : \{1, \ldots, n\} \leftrightarrow V$. $\alpha^{-1}(v)$ gives the order of $v$.

$G_{\alpha} = (V, E, \alpha)$ is an ordered graph.

Deficiency of a vertex: $D(v)$ is the set of edges defined by $D(v) = \{(x, y) : x \to v \text{ and } v \to y \text{ and } x \not\to y \text{ and } x \neq y\}$

$v$-elimination graph: Apply the elimination process to the vertex $v$ of $G$ to obtain $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v))$.

For an ordered graph $G_{\alpha} = (V, E, \alpha)$, the elimination process $P(G_{\alpha}) = [G = G_0, G_1, G_2, \ldots, G_{n-1}]$ is the sequence of elimination graphs defined by $G_0 = G$, $G_i = (G_{i-1})_{\alpha(i)}$.
Elimination process: Formal definitions

\[ P(G_\alpha) = [G = G_0, G_1, G_2, \ldots, G_{n-1}] \]
is the sequence of elimination graphs defined by \( G_0 = G, G_i = (G_{i-1})_{\alpha(i)} \). Let \( G_i = (V_i, E_i) \) for \( i = 0, 1, \ldots, n-1 \). The fill-in \( F(G_\alpha) \) is defined by

\[ F(G_\alpha) = \bigcup_{i=1}^{n-1} \tau_i \]

where \( \tau_i = D(\alpha(i)) \) in \( G_{i-1} \), and the elimination graph is defined by

\[ G^*_\alpha = (V, E \cup F(G_\alpha)) \]

For a matrix \( A \), \( \tau_i \) corresponds to the new nonzeros elements, the fill-ins, created during \( i \) the step of elimination.

Observe that \( (v, w) \in G^*_\alpha \), if \( (v, w) \in E \) or there is a vertex \( u \), \( \alpha^{-1}(u) < \min(\alpha^{-1}(v), \alpha^{-1}(w)) \) and both \( (u, v) \) and \( (u, w) \) are in \( G^*_\alpha \).
Elimination process: Formal definitions

Continuing from the previous sample matrix, we have the filled-graph $G^+(A)$

$$G^+(A) = G(F)$$

$$F = L + L^T$$
Elimination process: Formal definitions

Given a graph $G = (V, E)$, an ordering $\alpha$ of $V$ is a perfect elimination ordering of $G$ if $F(G_\alpha) = \emptyset$.

The ordering $\alpha$ is a perfect elimination ordering if $w \in \text{adj}(v)$, $x \in \text{adj}(v)$, and $\alpha^{-1}(v) < \min\{\alpha^{-1}(w), \alpha^{-1}(x)\}$ in $G_\alpha$, imply either $(w, x) \in E$ or $w = x$. In other words, when $v$ is to be eliminated (both $w$ and $x$ are not eliminated yet), there is an edge $(w, x)$.

A graph which has a perfect elimination ordering is a perfect elimination graph. Any elimination graph $G_\alpha^*$ is a perfect elimination graph, since $\alpha$ is a perfect ordering.

Similar statements for the directed graph case.
A graph $G$ is called **triangulated** if for every cycle $\mu = [v_1, v_2, \ldots, v_\ell]$ of length $\ell > 3$, there is an edge of $G$ joining two nonconsecutive vertices of $\mu$ (different names: chordal, monotone transitive, and rigid circuit graphs).
Elimination process: Perfect elimination

A graph $G$ is called **triangulated** if for every cycle $\mu = [v_1, v_2, \ldots, v_\ell]$ of length $\ell > 3$, there is an edge of $G$ joining two nonconsecutive vertices of $\mu$.

**Simplicial vertex:** is a vertex whose adjacency is a clique. In a triangulated graph, if not a clique, there are at least two nonadjacent simplicial vertices.

**Theorem:** $G$ is triangulated iff it has a perfect elimination ordering (Proof?). Any simplicial vertex can be the first vertex of a perfect elimination ordering.

**Lemma:** Let $\alpha$ be a perfect elimination ordering of a triangulated graph $G = (V, E)$ and let $x \in V$. Then, $\alpha$ is a perfect elimination ordering for $G' = (V, E \cup D(x))$. (Proof?).

**Corollary:** If $G = (V, E)$ is triangulated and $x$ any vertex, the elimination graph $G_x = (V - \{x\}, E(V - \{x\}) \cup D(x))$ is triangulated.
Elimination process: Perfect elimination

One can recognize triangulated graphs by a slightly modified breadth-first search algorithm called lexicographic breadth-first search (LexBFS).

assign the label $\emptyset$ to each vertex

for $i = n : -1 : 1$ do

pick an unnumbered vertex $v$ with the largest label (lexicographic)

$\alpha(i) \leftarrow v$

for each unnumbered vertex $w \in \text{adj}(v)$ do

$\text{label}(w) \leftarrow \text{label}(w) \oplus i$

We obtain an ordering $\alpha$. Graph is triangulated iff $\alpha$ is a perfect elimination ordering. The essential idea for such a test is to check at each step if the neighbors of a vertex (in the elimination graph) are equal to its original neighbors.

LexBFS runs in $\mathcal{O}(|V| + |E|)$ time to generate the ordering, and the ordering can be tested in $\mathcal{O}(|V| + |E'|)$ time to see if it is perfect (if so, then $E' = E$). (Try to write these algorithms).
Elimination process: LexBFS
Elimination process: Maximum cardinality search

Much simpler algorithm.

Number the vertices from \( n \) to 1 in decreasing order.

As the next vertex to number, select the vertex adjacent to the largest number of previously numbered vertices; break the ties arbitrarily.

What is the running time complexity? Is searching the next \( j \) a problem?

Maximum cardinality search.

```plaintext
local \( j, v; 
for i \in [0, n-1] \rightarrow set (i) := \emptyset \ rof; 
for v \in vertices \rightarrow size (v) := 0; \ add \ v \ to \ set (0) \ rof; 
i := n; j := 0;
\hspace{0.5cm} \text{do } i \geq 1 \rightarrow 
v := \text{delete any from set } (j); 
\alpha (v) := i; \alpha^{-1} (i) := v; \ size (v) := -1;
\hspace{0.5cm} \text{for } \{v, w\} \in E \text{ such that } size (w) \geq 0 \rightarrow 
\hspace{0.5cm} \text{delete } w \text{ from set } (size (w)); 
\hspace{0.5cm} size (w) := size (w) + 1;
\hspace{0.5cm} \text{add } w \text{ to set } (size (w)) 
\hspace{0.5cm} \rof; 
i := i - 1;
j := j + 1;
do j \geq 0 \text{ and set } (j) = \emptyset \rightarrow j := j - 1 \ \text{od} 
\text{od;}
```
Elimination process: What if the graph is not triangulated?

Fill-path theorem [Rose, Tarjan, Lueker’76]

Let \( G = (V, E, \alpha) \) be an ordered graph. Then \((v, w)\) is an edge of \( G^*_\alpha = (V, E \cup F(G_\alpha)) \) iff there exists a path \( \mu = [v = v_1, v_2, \ldots, v_{k+1} = w] \) in \( G \) such that

\[
\alpha^{-1}(v_i) < \min\{\alpha^{-1}(v), \alpha^{-1}(w)\}, \quad 2 \leq i \leq k
\]

In the graph of \( L + L^T \): \( \ell_{ij} \neq 0 \) iff there is a path in \( G(A, \alpha) \) with only vertices that come earlier than \( i \) and \( j \).

(Proof?)
Elimination process: What if the directed graph do not have peo?

Fill-path theorem [Rose, Tarjan,’78]

Let $G = (V, E, \alpha)$ be an ordered directed graph. Then $(v, w)$ is an edge of $G^\ast_\alpha = (V, E \cup F(G_\alpha))$ iff there exists a path

$\mu = [v = v_1, v_2, \ldots, v_{k+1} = w]$ in $G$ such that

$$\alpha^{-1}(v_i) < \min\{\alpha^{-1}(v), \alpha^{-1}(w)\}, \quad 2 \leq i \leq k$$

In the directed graph of $L + U$: $l_{ij} \neq 0$ iff there is a path in $G(A, \alpha)$ with only vertices that come earlier than $i$ and $j$ (and $j$ comes earlier than $i$). Similar for the upper triangular part.
How significant is this?

\[ \times \times \times \times \times \]
\[ \times \times \]
\[ \times \times \]
\[ \times \times \]
\[ \times \times \]
For undirected graphs, we can recognize in $O(n + \tau)$ time if the given graph has a peo.

What about the directed graphs?
Question: LU and reducibility

Prove or disprove:

Let $A$ be a nonsingular matrix with $A = LU$ and irreducible.

Then every column in $L$ has a nonzero below the diagonal.

Then every row in $U$ has a nonzero to the right of the diagonal.
Question: LU and reducibility

Prove or disprove:

Let \( A \) be a nonsingular matrix with \( A = LU \) and irreducible.

Then every column in \( L \) has a nonzero below the diagonal.

Then every row in \( U \) has a nonzero to the right of the diagonal.

Hint: Use the directed graph version of the fill-path theorem.
Question

When is the inverse of a sparse matrix is sparse/full?
Question

When is the inverse of a sparse matrix is sparse/full?

**Hint:** assume $A = LU$ is an LU factorization and consider obtaining the inverse with solves.
Question

When is the inverse of a sparse matrix is sparse/full?
reducible, irreducible...
Suppose $W$ is a connected subgraph of $G$ and $\alpha$ is an elimination ordering.
Let $Z = \{v : v \in \text{adj}(w) \text{ for some } w \in W \text{ and } \alpha^{-1}(v) > \max_{u \in W} \alpha^{-1}(u)\}$
Then $Z \cup u$ where $u \in W$, and $\alpha^{-1}(u)$ is the largest in $W$ is a clique at the time we eliminate $u$. 
Elimination process (undirected graphs): What if the graph is not triangulated?

Ideally, we would like to find an ordering with minimum fill-in. However, the minimum fill-in problem is NP-complete (Yannakakis,’81)

Proof. \(\langle\langle\text{On the board}\rangle\rangle\)

Definitions: A bipartite graph \(G = (P, Q, E)\) is a chain graph if there is an ordering \(\pi\) of \(P\) such that \(\text{adj}(\pi(1)) \supseteq \text{adj}(\pi(2)) \supseteq \cdots \supseteq \text{adj}(\pi(|P|))\). It is a hereditary property. Such an ordering in \(Q\) also exists. It is NP-complete to find the minimum number of edges whose addition to a bipartite graph gives a chain graph.

Two edges \((u, v)\) and \((x, y)\) are independent if no other edges between these four vertices. A bipartite graph is a chain graph iff it does not contain a pair of independent edges.

Reduction from chain graph completion.

Hint: Given a bipartite graph, \(G = (P, Q, E)\) consider the graph \(C(G) = (P \cup Q, E')\) where \(E' = E \cup \{(a, b): a, b \in P\} \cup \{(c, d): c, d \in Q\}\). Show that \(C(G)\) is chordal iff \(G\) is a chain graph.
Elimination process (undirected graphs): What if the graph is not triangulated?

Ideally, we would like to find an ordering with minimum fill-in. However, the minimum fill-in problem is NP-complete (Yannakakis,'81) and hence heuristics are used.

What else we may want to optimize?

- operation count during elimination,
- parallel performance,
- memory consumption.
Outline

1. Sparse triangular system solutions
2. Predicting the structure of factorization
   - Elimination process on the graphs
3. Ordering sparse matrices
   - Fill-reducing orderings
   - Local fill-reducing ordering methods
   - Minimum fill-in heuristic
   - Global fill-reducing ordering methods
4. Closing
A is symmetric positive definite; diagonals are $> 0$ even during elimination $> 0$.

We can choose any diagonal entry safely and use it as a pivot. Since all are safe, we can choose the pivot which brings less fill-in. We can avoid the search during factorization and do an ordering beforehand.
Ordering example

Original matrix

Reordered matrix
Fill-reducing ordering methods

Three main classes of methods for minimizing fill-in during factorization

- Local approaches: At each step of the factorization, selection of the pivot that is likely to minimize fill-in.
  - Method is characterized by the way pivots are selected.
  - Markowitz criterion (for a general matrix) [Markowitz,'57].
  - Minimum degree (for symmetric matrices) [Tinney and Walker,'67], [Rose,'72].

\[
G_U(V, E) \leftarrow \text{undirected graph of } A \\
\text{for } i = 1 : n - 1 \text{ do} \\
\quad \text{let } k \text{ be a vertex that minimizes a metric} \\
\quad V \leftarrow V - \{k\} \triangleright \text{remove vertex } k \\
\quad E \leftarrow E - \{(k, \ell) : \ell \in \text{adj}(k)\} \cup \{(x, y) : x \in \text{adj}(k) \text{ and } y \in \text{adj}(k)\}
\]
Fill-reducing ordering methods

- Global approaches: The matrix is permuted so as to confine the fill-in within certain parts of the permuted matrix
  - Cuthill-McKee [Cuthill and McKee, ’69], Reverse Cuthill-McKee [George, ’71]
  - Nested dissection [George, ’73]

- Hybrid approaches: First permute the matrix globally to confine the fill-in, then reorder small parts using local heuristics.
Local heuristics to reduce fill-in during factorization

Let $G(A)$ be the graph associated to a matrix $A$ that we want to order using local heuristics.

Let $Metric$ such that $Metric(v_i) < Metric(v_j)$ implies $v_i$ is better than $v_j$.

---

**Generic algorithm**

Loop until all nodes are selected

1: select current node $p$ (so called pivot) with minimum metric value,
2: update elimination graph,
3: update $Metric(v_j)$ for all non-selected nodes $v_j$.

3rd step should only be applied to nodes for which the Metric value might have changed.
Reordering unsymmetric matrices: Markowitz criterion

- At step $k$ of Gaussian elimination:

  ![Diagram showing Gaussian elimination process]

  - $r_i^{(k)} = \text{number of non-zeros in row } i \text{ of } A^{(k)}$
  - $c_j^{(k)} = \text{number of non-zeros in column } j \text{ of } A^{(k)}$
  - Candidate pivot $a_{ij}$ must be large enough and should minimize $(r_i^{(k)} - 1) \times (c_j^{(k)} - 1)$ for all $i, j \geq k$

- Minimum degree: Said to be Markowitz criterion for symmetric diagonally dominant matrices
Minimum degree algorithm

- **Step 1:**
  Select the vertex that possesses the smallest number of neighbors in $G^0$.

\[
\begin{bmatrix}
1 & x & x & x \\
2 & x & x & x & x \\
3 & x & x & x \\
x & 4 & x & x \\
x & x & 5 & x & x \\
x & x & x & 6 \\
x & x & x & x & 7 & x & x & x \\
x & x & x & x & 8 & x & x \\
x & x & x & x & x & 9 & x \\
x & x & x & x & x & x & 10
\end{bmatrix}
\]

(a) Sparse symmetric matrix

(b) Elimination graph

The node/variable selected is 1 of degree 2.
Step 1: elimination of pivot 1

(a) Elimination graph

(b) Factors and active submatrix

Initial nonzeros Fill−in Nonzeros in factors
Minimum degree algorithm based on elimination graphs

\[ \forall i \in [1 \cdots n] \quad t_i = |\text{adj} G_0(i)| \]

\textbf{For} \quad k = 1 \text{ to } n \text{ \textbf{Do}}

\[ p = \min_{i \in V_{k-1}} (t_i) \]

\textbf{For} \quad \text{each } i \in \text{adj} G_{k-1}(p) \text{ \textbf{Do}}

\[ \text{adj} G_k(i) = (\text{adj} G_{k-1}(i) \cup \text{adj} G_{k-1}(p)) \setminus \{i, p\} \]

\[ t_i = |\text{adj} G_k(i)| \]

\textbf{EndFor}

\[ V^k = V^{k-1} \setminus p \]

\textbf{EndFor}
Illustration (cont’d)

Graphs $G_1, G_2, G_3$ and corresponding reduced matrices.

(a) Elimination graphs

(b) Factors and active submatrices

× Original nonzero
☑ Original nonzero modified
■ Fill-in
● Nonzeros in factors
Minimum Degree does not always minimize fill-in.

Consider the following matrix

\[
\begin{bmatrix}
1 & x & x & x \\
x & 2 & x & x \\
x & x & 3 & x \\
x & x & x & 4 \\
x & x & x & 5 \\
x & x & x & 6 \\
x & x & x & x \\
x & x & 8 & x \\
& x & x & 9
\end{bmatrix}
\]

Remark: Using initial ordering
No fill-in

Step 1 of Minimum Degree:
Select pivot 5 (minimum degree = 2)
Updated graph

Add (4, 6) i.e. fill-in
Minimum degree ordering: What more to say?

Some improvements and algorithmic follow-ups

- Tie breaking [Duff, Erisman, and Reid,'76],
- Mass eliminations [George and McIntyre,'78], Indistinguishable nodes [George and Liu,'81]
- Use of quotient graphs; $O(\text{nnz}(A))$ space [George and Liu,'80],
- Element absorption, [Duff and Reid,'83],
- Multiple minimum degree, [Liu,'85],
- Minimum degree with constraints [Liu,'89],
- Approximate minimum degree [Amestoy, Davis, and Duff,'96].

Open-ended

As the problem is **NP-complete**, and as the minimum degree algorithm depends highly on the tie-breaking mechanisms, there is always scope for more work.
Minimum degree ordering: Running time

\[ O(n^2m) \] for exact, and \[ O(nm) \] for approximate.
Influence on the structure of factors

Harwell-Boeing matrix: dwt_592.rua, structural computing on a submarine. NZ(LU factors) = 58202
Detection of *supervariables* allows to build more regularly structured factors (easier factorization).
Recalling the generic algorithm

Let $G(A)$ be the graph associated to a matrix $A$ that we want to order using local heuristics.

Let $Metric$ be such that $Metric(v_i) < Metric(v_j) \equiv v_i$ is better than $v_j$

**Generic algorithm**

Loop until all nodes are selected
1. Select current node $p$ (so called pivot) with minimum metric value,
2. update elimination graph,
3. update $Metric(v_j)$ for all non-selected nodes $v_j$.

3rd step should only be applied to nodes for which the Metric value might have changed.
Minimum fill based algorithm

- Metric($v_i$) is the amount of fill-in that $v_i$ would introduce if it were selected as a pivot.
- Illustration: $r$ has a degree $d = 4$ and a fill-in metric of $d \times (d - 1)/2 = 6$ whereas $s$ has degree $d = 5$ but a fill-in metric of $d \times (d - 1)/2 - 9 = 1$. 
CM and RCM: Definitions

Bandwidth: A structurally symmetric matrix $A$ is said to have bandwidth $2m + 1$, if $m$ is the smallest integer such that $a_{ij} = 0$, whenever $|i - j| > m$. If no interchanges are performed during elimination, fill-in occurs only within the band. It is NP-complete to minimize the bandwidth of a symmetric matrix, by symmetric permutations. What about unsymmetric matrices?

Profile: Define bandwidth for each row $i$: $m(i)$ is the smallest integer such that $a_{ij} = 0$, whenever $i - j > m(i)$ for $j < i$. Then profile of a symmetric matrix is $\sum_i m(i)$. If no interchanges are performed during elimination, no fill-in occurs ahead of the first entry in each row.

Block tridiagonal form: Nonzeros are on the diagonal blocks or in a block just above the diagonal or just below the diagonal.
Level sets (as in breadth-first search) are built from the vertex of minimum degree. At a level, priority is given to a vertex with smaller number of neighbors.

pick a vertex $v$ and order it as the first vertex

$S \leftarrow \{v\}$

while $S \neq V$ do

$S' \leftarrow$ all vertices in $V \setminus S$ which are adjacent to $S$

order vertices in $S'$ in increasing order of degrees

$S \leftarrow S \cup S'$
CM and RCM: Algorithm

(from Duff, Erisman, Reid)
CM vs RCM

CM ordering

RCM ordering

$A_c = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix}$

$A_r = \begin{bmatrix} x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \\ x & x & x & x & x & x \end{bmatrix}$

RCM: Simply reverse the order found by the CM algorithm.

It does not change the bandwidth but improves the storage requirements.

(Liu and Sherman,’76 )
Illustration: Reverse Cuthill-McKee on matrix dwt_592

Harwell-Boeing matrix: dwt_592, structural computing on a submarine.
NZ(LU factors)=58202
Illustration: Reverse Cuthill-McKee on matrix dwt_592

\[ \text{NZ}(\text{LU factors}) = 16924 \]

**Permuted matrix (RCM)**

**Factorized permuted matrix**
Nested dissection

Originally proposed by George,'73 for finite-element meshes.

It gives time and space bounds for linear system solutions associated with such meshes (both 2D and 3D). In 2D $O(n \log n)$ space and $O(n^{3/2})$ time (here $n$ is the total number of points).

Was generalized to systems defined on planar or almost planar graphs by Lipton, Rose, and Tarjan,'79 with the same bounds on time and space.

George and Liu,'78 use level sets on undirected graphs to automate the ND of a sparse symmetric matrix (We will cover more sophisticated methods later in the semester).

Gilbert and Tarjan,'87 extend the result to some other graph classes.
ND of a regular square mesh

Structure of $L$

(from Cleve Ashcraft)
Nested dissection

A $1138 \times 1138$ matrix from UFL sparse matrix collection, with 4045 nonzeros.

- $\text{nnz}(L) = 38312$
- $\text{nnz}(L_{RCM}) = 5222$
- $\text{nnz}(L_{ND}) = 4287$
George, ’73 shows that $O(n \log n)$, for a $k \times k$ mesh with $n = k^2$.

Hoffman, Martin, and Rose show that the model problem has $\Omega(n \log n)$ fill.
Outline

1. Sparse triangular system solutions

2. Predicting the structure of factorization
   - Elimination process on the graphs

3. Ordering sparse matrices
   - Fill-reducing orderings
   - Local fill-reducing ordering methods
   - Minimum fill-in heuristic
   - Global fill-reducing ordering methods

4. Closing
Use Cholesky to solve $Ax = b$

1. Find a good ordering $P$, replace $A$ by $PAP^T$, and $b$ by $Pb$.
2. Factorize $A = LL^T$.
3. Triangular solves $Ly = b$, $L^Tz = y$, return $P^Tz$ as $x$
Could not get enough of it: Questions

In what order would you eliminate the vars of a system whose matrix is

- a path
- a tree
- a ladder
- a square mesh
- a rectangular mesh

Ideas?

We talked about minimum degree, minimum fill-in metrics. What about the minimum operation count metric?