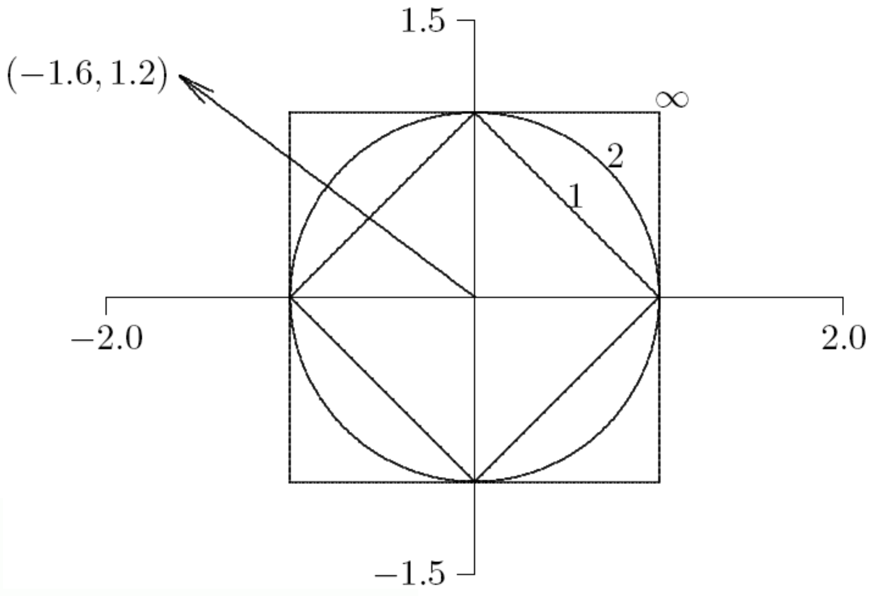
Linear System

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$



Important special cases

- 1-norm: $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$
- 2-norm: $\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$
- ∞-norm: $\|\boldsymbol{x}\|_\infty = \max_i |x_i|$

Matrix norm corresponding to given vector norm is defined by

$$\|\boldsymbol{A}\| = \max_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}$$

Condition number of square nonsingular matrix $\boldsymbol{A}$ is defined by

$$\operatorname{cond}(\boldsymbol{A}) = \|\boldsymbol{A}\| \cdot \| \boldsymbol{A}^{-1}\|$$

$$\|\boldsymbol{A}\| \cdot \| \boldsymbol{A}^{-1}\| = \left(\max_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\right) \cdot \left(\min_{\boldsymbol{x}\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\right)^{-1}$$

condition number measures ratio of maximum stretching to maximum shrinking matrix does to any nonzero vectors

Case 1:

Let x be the solution of Ax=b and $\hat{x}$ be the solution of $A\hat{x}=b+\Delta b$, the error $\Delta x = x - \hat{x}$ and the residual $r = b - A\hat{x} = A\left(x - \hat{x}\right)$:

we have $\|\Delta x\| \le \left\|A^{-1}\right\|\|r\|$ together with $b = Ax$

$\Rightarrow \|b\| \le \|A\|\|x\|$, the inequality $\dfrac{\|\Delta x\|}{\|x\|} \le cond\left(A\right)\dfrac{\|r\|}{\|b\|}$

- Small residual is easy to obtain, but does not necessarily imply computed solution is accurate
- small relative residual implies small relative error in approximate solution *only if A is well-conditioned*

Case 2:

Let $\hat{x}$ be the solution of $(A+E)\hat{x}=b$, the error $\Delta x = x - \hat{x}$ and the residual $r = b - A\hat{x} = A\left(x - \hat{x}\right)$:

- Similar result holds for relative change in matrix: if $(\boldsymbol{A} + \boldsymbol{E})\hat{\boldsymbol{x}} = \boldsymbol{b}$, then

$$\frac{\|\Delta \boldsymbol{x}\|}{\|\hat{\boldsymbol{x}}\|} \leq \operatorname{cond}(\boldsymbol{A}) \frac{\|\boldsymbol{E}\|}{\|\boldsymbol{A}\|}$$

- If input data are accurate to machine precision, then bound for relative error in solution $\boldsymbol{x}$ becomes

$$\frac{\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \leq \operatorname{cond}(\boldsymbol{A})\, \epsilon_{\text{mach}}$$

$$\|r\| = \|b - A\hat{x}\| = \left\|A(x - \hat{x})\right\| \le \|A\|\|\Delta x\|$$

$$\|x\| \le \left\|A^{-1}\right\|\|b\|$$

$$\frac{\|r\|}{\left\|A^{-1}\right\|\|b\|} \le \frac{\|A\|\|\Delta x\|}{\|x\|} \Rightarrow \frac{\|r\|}{\|b\|} \le cond(A)\frac{\|\Delta x\|}{\|x\|}$$

- For well-conditioned A, large *relative residual implies large backward error in* matrix, and algorithm used to compute solution is unstable.

- For ill-conditioned A, large relative residual does not necessary imply the relative error is also large.

$$E\hat{x} = b - A\hat{x} = r \Rightarrow \|r\| \le \|E\|\|\hat{x}\| \le \|E\|\left\|(A+E)^{-1}\right\|\|b\|$$

$$\Rightarrow \frac{\|r\|}{\|b\|} \le \|E\|\left\|\left(A\left(I + A^{-1}E\right)\right)^{-1}\right\| \le \|E\|\|A^{-1}\|\left\|\left(I + A^{-1}E\right)^{-1}\right\|$$

$$< \|E\|\|A^{-1}\|\frac{1}{1\text{-}\|A^{-1}E\|} \text{ when } \|A^{-1}E\| << 1 \text{ (see Lemma 1 at p.30)}$$

$$\approx \frac{\|E\|}{\|A\|} cond(A)$$

One can estimate the backward error is about

$$\frac{\|\Delta x\|}{\|x\|} \approx O\left(\frac{\|E\|}{\|A\|}\right)$$

Solve linear system by iterations

Direct Solver: Gauss Elimination based on LU decomposition:

Observation (1)

- *Forward-substitution* for lower triangular system $Lx = b$

$$x_1 = b_1/\ell_{11}, \quad x_i = \left(b_i - \sum_{j=1}^{i-1} \ell_{ij} x_j \right) / \ell_{ii}, \quad i = 2, \ldots, n$$

for $j = 1$ **to** n	{ loop over columns }
if $\ell_{jj} = 0$ **then** stop	{ stop if matrix is singular }
$x_j = b_j/\ell_{jj}$	{ compute solution component }
for $i = j + 1$ **to** n	
$b_i = b_i - \ell_{ij} x_j$	{ update right-hand side }
end	
end	

Observation (2):

- *Back-substitution* for upper triangular system $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$

$$x_n = b_n / u_{nn}, \quad x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / \, u_{ii}, \quad i = n-1, \ldots, 1$$

for $j = n$ **to** 1 { loop backwards over columns }
  if $u_{jj} = 0$ **then** stop { stop if matrix is singular }
  $x_j = b_j / u_{jj}$ { compute solution component }
  for $i = 1$ **to** $j - 1$
    $b_i = b_i - u_{ij} x_j$ { update right-hand side }
  end
end

Question (1)

Can one decompose a matrix A into the product of a lower triangular matrix L and a upper triangular matrix U?

Question (2)

Suppose we can, what would be the algorithm to find L and U?

Question (3)

Is the algorithm in question (2) stable?

Existence of LU-Decomposition

LU decomposition exists when all leading principal minors of the n x n matrix A are nonsingular.

i.e.

$$A^{(k)} = \begin{pmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{pmatrix}, \ k = 1 \sim n.$$

This is difficult to check in real computation.

Every strictly diagonally dominant matrix is nonsingular and has an LU-factorization.

Strictly diagonal-dominant matrix $\left|a_{ii}\right| > \sum_{j \neq i} \left|a_{ij}\right|$, for all i

Proof: Consider

$$A = \begin{bmatrix} \alpha & \omega^T \\ \upsilon & C \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \beta & \mathrm{I} \end{bmatrix} \begin{bmatrix} \alpha & \gamma \\ 0 & \delta \end{bmatrix} \Rightarrow \begin{cases} \gamma = \omega^T \\ \beta\alpha = \upsilon \ \Rightarrow \beta = \dfrac{\upsilon}{\alpha} \\ \delta = C - \beta\gamma = C - \dfrac{\omega^T \upsilon}{\alpha} \end{cases}$$

$$\Rightarrow A = \begin{bmatrix} 1 & 0 \\ \dfrac{\upsilon}{\alpha} & \mathrm{I} \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & C - \dfrac{\upsilon\omega^T}{\alpha} \end{bmatrix} \begin{bmatrix} \alpha & \omega^T \\ 0 & \mathrm{I} \end{bmatrix}}_{\substack{u \\ \begin{bmatrix} \alpha & \omega^T \\ 0 & C - \frac{\upsilon\omega^T}{\alpha} \end{bmatrix}}}$$

If one can show $B = C - \dfrac{\omega^T \upsilon}{\alpha}$ is diagonally dominant, when

A and C are diagnoally dominant then, by assumption of induction,

we have $B = \tilde{L}\tilde{U}$

$$\Rightarrow A = \underbrace{\begin{bmatrix} 1 & 0 \\ \dfrac{\upsilon}{\alpha} & \mathrm{I} \end{bmatrix} \cdot \tilde{L}}_{L} \cdot \underbrace{\tilde{U} \cdot \begin{bmatrix} \alpha & \omega^T \\ 0 & \mathrm{I} \end{bmatrix}}_{U} \quad \Rightarrow \text{By math induction,}$$

A has a L-U factorization. Finally, since

$$\sum_{\substack{j=1 \\ i \neq j}}^{n\text{-}1} |b_{ij}| = \sum_{\substack{j=1 \\ i \neq j}}^{n\text{-}1} \left| c_{ij} - \frac{\upsilon_i \omega_j}{\alpha} \right| \leq \sum_{\substack{j=1 \\ i \neq j}}^{n\text{-}1} |c_{ij}| + \left| \frac{\upsilon_i}{\alpha} \right| \sum_{\substack{j=1 \\ i \neq j}}^{n\text{-}1} |\omega_j|$$

$$< |c_{ii}| - |\upsilon_i| + \left| \frac{\upsilon_i}{\alpha} \right| \Big(\underbrace{\sum_{j=1}^{n-1} |\omega_j|}_{<|\alpha|} - |\omega_i| \Big) < |c_{ii}| - \frac{|\upsilon_i||\omega_i|}{\alpha}$$

$$\leq \left| c_{ii} - \frac{\upsilon_i \omega_i}{\alpha} \right| = |b_{ii}|,$$

Hence, B is strictly diagnoally dominant.

Remark:

- If A is irreducibly, diagonally dominant with strictly inequality holds for at least one row, then A is non-singular and has a LU factorization.

(assuming the strict dominant inequality. holds at first row [α , ω^T], clearly, the above argument still holds.)

- LU factorization is good for multiple right hand sides.

- LU uniqueness:

  - Despite variations in computing it, LU factorization is unique up to diagonal scaling of factors
  - Provided row pivot sequence is same, if we have two LU factorizations $PA = LU = \hat{L}\hat{U}$, then $\hat{L}^{-1}L = \hat{U}U^{-1} = D$ is both lower and upper triangular, hence diagonal
  - If both L and $\hat{L}$ are unit lower triangular, then D must be identity matrix, so $L = \hat{L}$ and $U = \hat{U}$
  - Uniqueness is made explicit in LDU factorization $PA = LDU$, with L unit lower triangular, U unit upper triangular, and D diagonal

LU Algorithm

For $k = 1 : n,$

$$\ell_{kk} u_{kk} = a_{kk} - \sum_{m=1}^{k-1} \ell_{km} u_{mk}$$

For $j = k+1 : n,$

$$u_{kj} = \underbrace{(a_{kj} - \sum_{m=1}^{k-1} \ell_{km} u_{mj})}_{\text{2k-1 operations}} / \ell_{kk}$$

end.

For $i = k+1 : n,$

$$\ell_{ik} = \underbrace{(a_{ik} - \sum_{m=1}^{k-1} \ell_{im} u_{mj})}_{\substack{\text{(k-1)+(k-2)+1+1=2k-1} \\ \text{operations}}} / u_{kk}$$

end

end

Computation cost

$$2\left(\sum_{k=1}^{n} (n-k) \cdot (2k-1)\right) +$$

$$\sum_{k=1}^{n} 2(k-1) = \mathrm{O}(n^3)$$

Remark:

- $O\left(n^3\right)$ computational cost is too expensive.
- Possible halt when l_{kk} or $u_{kk} = 0,$ pivoting strategy is needed.

Consider A= $\begin{bmatrix} \varepsilon & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & 1 \\ 0 & 1-\frac{1}{\varepsilon} \end{bmatrix} = L \cdot U$. When overflow,

$L \cdot U \approx \begin{bmatrix} 1 & 0 \\ \frac{1}{\varepsilon} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & 1 \\ 0 & -\frac{1}{\varepsilon} \end{bmatrix} = \begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix} = \hat{A},$ one has $\|\mathrm{E}\| = \left\|A - \hat{A}\right\| = O(1),$

according to the backward error estimation

$$\frac{\|\Delta x\|}{\|x\|} \approx O\left(\frac{\|E\|}{\|A\|}\right) \approx O(1)$$

The solution is unreliable with about 100% relative error exercise: check the estimate by solving $\mathrm{Ax}=\begin{bmatrix}1\\0\end{bmatrix}$ and $\hat{\mathrm{A}}\hat{\mathrm{x}}=\begin{bmatrix}1\\0\end{bmatrix}$, and computing the error.

Pivoting strategy: largest entries should be ordered first

$Ax=b \;\Rightarrow\; PAx=Pb \;\Rightarrow\; A_p x=b_p$. Consider

$$A_p=\underbrace{\begin{bmatrix}0&1\\1&0\end{bmatrix}}_{P}\underbrace{\begin{bmatrix}\varepsilon&1\\1&1\end{bmatrix}}_{A}=\begin{bmatrix}1&1\\\varepsilon&1\end{bmatrix}=\begin{bmatrix}1&0\\\varepsilon&1\end{bmatrix}\begin{bmatrix}1&1\\0&1-\varepsilon\end{bmatrix}.$$

Even when underflow occurs in evaluating 1-ε,

$$\hat{A}_p=\begin{bmatrix}1&0\\\varepsilon&1\end{bmatrix}\begin{bmatrix}1&1\\0&1\end{bmatrix}=\begin{bmatrix}1&1\\\varepsilon&1+\varepsilon\end{bmatrix}\Rightarrow\left\|A_p-\hat{A}_p\right\|=O(\varepsilon).$$

So, the solution from the pivoted system is reliable.

Example:

(Hydraulic network) Let us consider the hydraulic network shown in the right figure, which is fed by a reservoir of water at constant pressure *pr = 10 bar. In this problem, pressure values* refer to the difference between the real pressure a the atmospheric one.

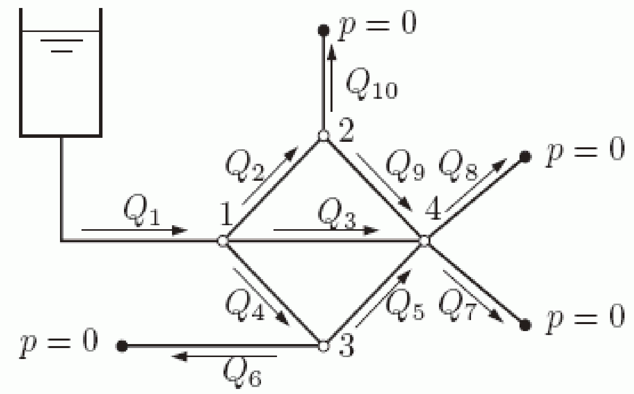


Fig. 5.1. The pipeline network of Problem 5.1

For the *j-th pipeline, the following relationship holds between the* flow-rate *Qj and the pressure gap* Δpj *at pipe-ends: Qj = kL*Δ*pj* , where *k is the hydraulic resistance and L is the length* of the pipeline. We assume that water flows from the outlets at atmospheric pressure, which is set to 0 bar. What is the pressure values at each internal node 1, 2, 3, 4 ?

pipeline	k	L	pipeline	k	L	pipeline	k	L
1	0.01	20	2	0.005	10	3	0.005	14
4	0.005	10	5	0.005	10	6	0.002	8
7	0.002	8	8	0.002	8	9	0.005	10
10	0.002	8						

Answer

$$Q_2 + Q_3 + Q_4 = Q_1$$
$$Q_9 + Q_{10} = Q_2$$
$$Q_9 + Q_3 + Q_5 = Q_7 + Q_8$$
$$Q_5 + Q_6 = Q_4$$

$$\Rightarrow \quad k_2 L_2 (P_2 - P_1) + k_3 L_3 (P_4 - P_1) + k_4 L_4 (P_3 - P_1) = k_1 L_1 (P_1 - 10)$$

$$\Downarrow$$

$$0.005 \times 10 \times (P_2 - P_1) + 0.005 \times 14 \times (P_4 - P_1) + 0.005 \times 10 \times (P_3 - P_1) =$$
$$0.01 \times 20 \times (P_1 - 10)$$

$$\Downarrow$$

$$-0.37 P_1 + 0.05 P_2 + 0.05 P_3 + 0.07 P_4 = -2$$

$$\mathrm{A} = \begin{bmatrix} -0.370 & 0.050 & 0.050 & 0.070 \\ \hline 0.050 & -0.116 & 0 & 0.050 \\ 0.050 & 0 & -0.116 & 0.050 \\ 0.070 & 0.050 & 0.050 & -0.202 \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} -2 \\ \hline 0 \\ 0 \\ 0 \end{bmatrix}.$$

Rank one updating

Sherman-Morrison Formula

- Sometimes refactorization can be avoided even when matrix *does* change
- *Sherman-Morrison formula* gives inverse of matrix resulting from rank-one change to matrix whose inverse is already known

$$(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1}$$

  where $\boldsymbol{u}$ and $\boldsymbol{v}$ are n-vectors

- Evaluation of formula requires $\mathcal{O}(n^2)$ work (for matrix-vector multiplications) rather than $\mathcal{O}(n^3)$ work required for inversion

- To solve linear system $(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)\boldsymbol{x} = \boldsymbol{b}$ with new matrix, use Sherman-Morrison formula to obtain

$$\begin{aligned} \boldsymbol{x} &= (\boldsymbol{A} - \boldsymbol{u}\boldsymbol{v}^T)^{-1}\boldsymbol{b} \\ &= \boldsymbol{A}^{-1}\boldsymbol{b} + \boldsymbol{A}^{-1}\boldsymbol{u}(1 - \boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{u})^{-1}\boldsymbol{v}^T\boldsymbol{A}^{-1}\boldsymbol{b} \end{aligned}$$

  which can be implemented by following steps
  - Solve $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{u}$ for $\boldsymbol{z}$, so $\boldsymbol{z} = \boldsymbol{A}^{-1}\boldsymbol{u}$
  - Solve $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$, so $\boldsymbol{y} = \boldsymbol{A}^{-1}\boldsymbol{b}$
  - Compute $\boldsymbol{x} = \boldsymbol{y} + ((\boldsymbol{v}^T\boldsymbol{y})/(1 - \boldsymbol{v}^T\boldsymbol{z}))\boldsymbol{z}$
- If $\boldsymbol{A}$ is already factored, procedure requires only triangular solutions and inner products, so only $\mathcal{O}(n^2)$ work and no explicit inverses

Rank m update

Sherman-Morrison-Woodbury Formula

$$\left(\mathrm{A}+\mathrm{U}\mathrm{V}^{\mathrm{T}}\right)^{-1} = A^{-1} - A^{-1}U\left(I + V^{T}A^{-1}U\right)^{-1}V^{T}A^{-1}$$

where U and V are $n \times m$ matrices, $n >> m$.

Exercise:

Use the LU algorithm to solve the equation $\begin{bmatrix} 4 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Use rank one updating to solve the equation $\begin{bmatrix} 4 & -1 & 0 & -1 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ -1 & 0 & -1 & 4 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

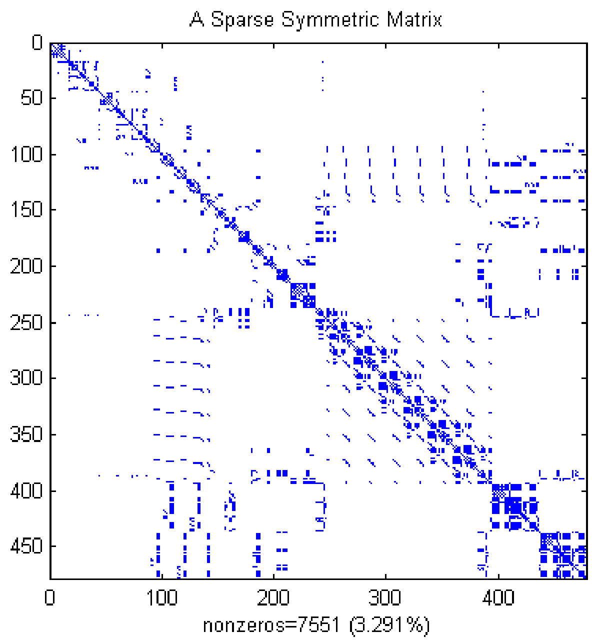
Possible fill-in in LU might give high storage cost :

Node reordering algorithm can improve !!! A classical algorithm is the Cuthill Mckee reordering

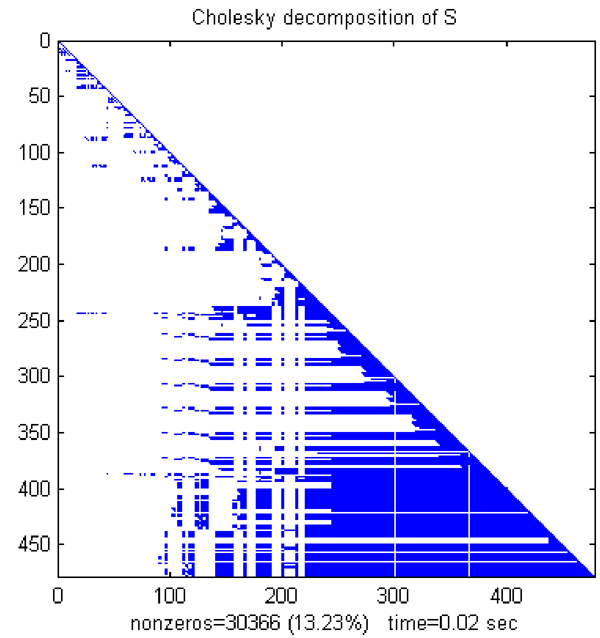
➤ **Difficulty in Gaussian elimination: Fill-in**

Trivial Example:

$$A = \begin{pmatrix} + & + & + & + & + & + \\ + & + & & & & \\ + & & + & & & \\ + & & & + & & \\ + & & & & + & \\ + & & & & & + \end{pmatrix}$$

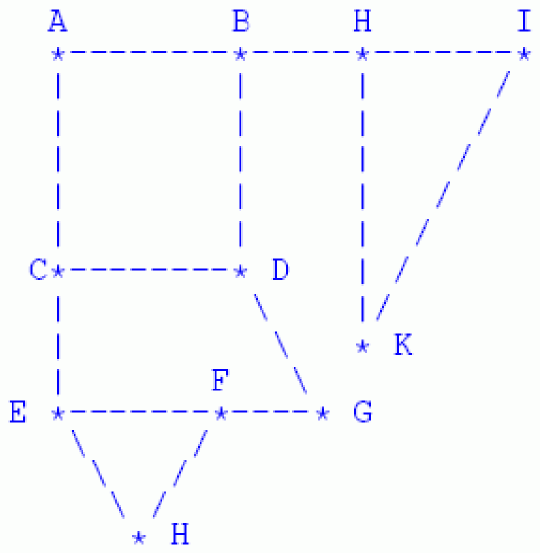


7550 non-zero elements



30366 non-zero elements

Breadth First Search




```
BFS from node A:
Level 0: A
Level 1: B, C;
Level 2: E, D, H;
Level 3: I, K, E, F, G, H.
```

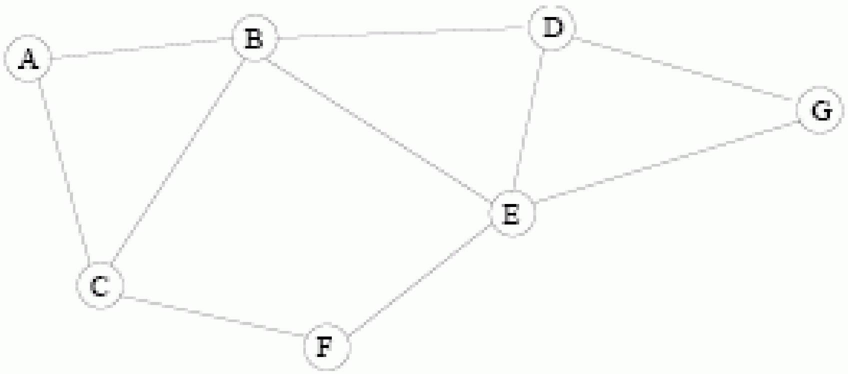
Algorithm $BFS(G, v)$ – by level sets –

- Initialize $S = \{v\}$, $seen = 1$; Mark v;
- While $seen < n$ Do
  - $S_{new} = \emptyset$;
  - For each node v in S do
    - For each unmarked w in adj(v) do
      - Add w to S_{new};
      - Mark w;
      - $seen++$;
  - $S := S_{new}$

Cuthill McKee ordering

Algorithm proceeds by levels. Same as BFS except: in each level, nodes are ordered by increasing degree

Example



Level	Nodes	Deg.	Order
0	A	2	A
1	B, C	4, 3	C, B
2	D, E, F	3, 4, 2	F, D, E
3	G	2	G

ALGORITHM : 1 . *Cuthill Mc Kee ordering*

Find an intial node for the traversal

Initialize $S = \{v\}, seen = 1, \pi(seen) = v$; *Mark* v;

While $seen < n$ *Do*

    $S_{new} = \emptyset$;

    For each node v, *going from lowest to highest degree, Do:*

        $\pi(++seen) = v$;

        For each unmarked w *in adj(*v*) do*

            Add w *to* S_{new};

            Mark w;

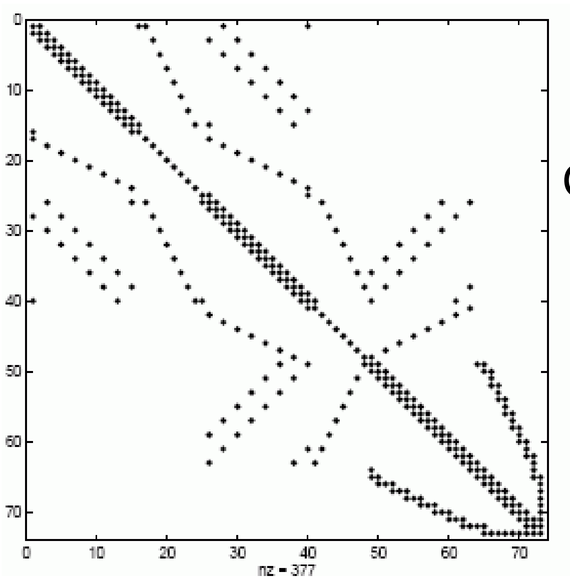
        EndDo

        $S := S_{new}$

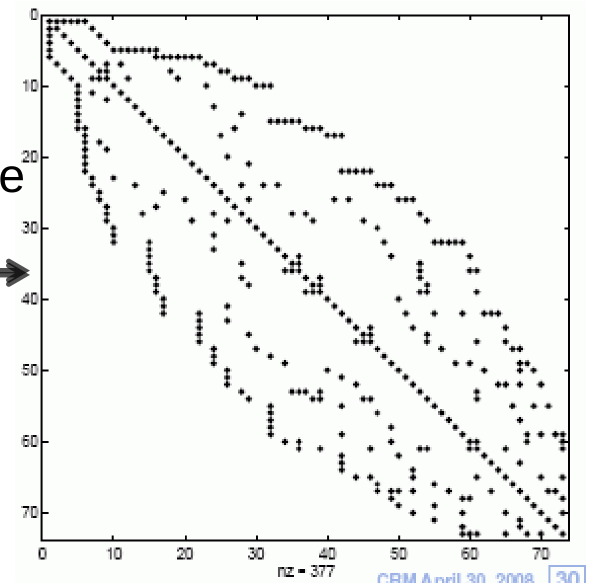
    EndDo

EndWhile

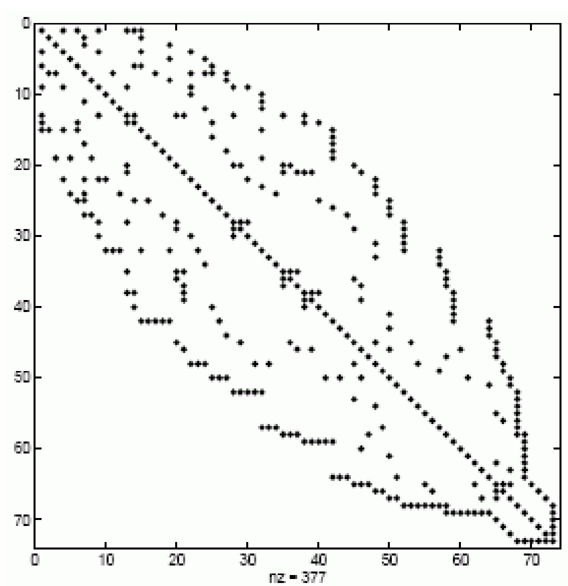
Reverse Cuthill Mckee : reverse the Cuthill-Mckee order

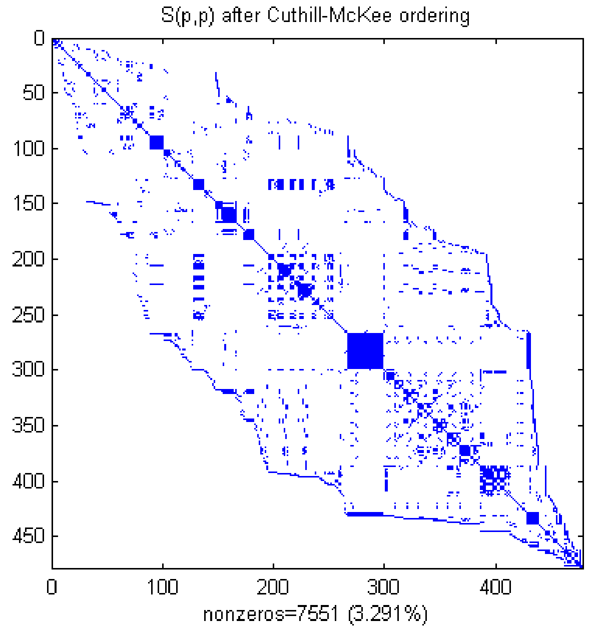


Cuthill-Mckee

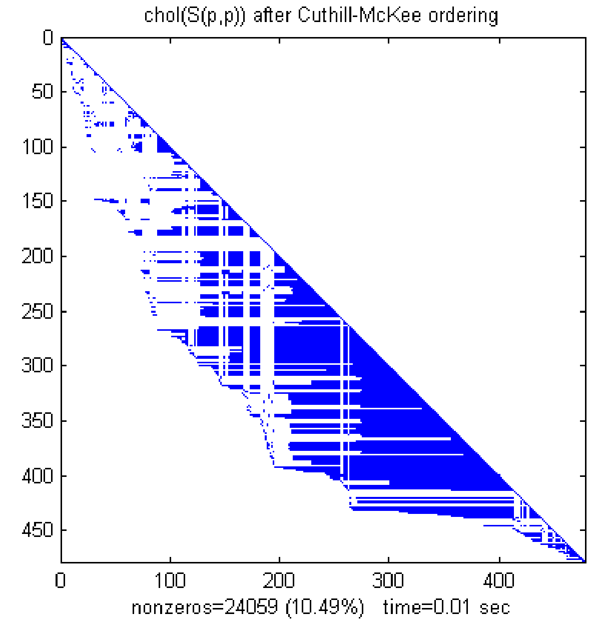


Reverse order





After reverse Cuthill_Mckee



24059 nonzero in LU

There are other reordering algorithm available such as column count and minimal degree reordering, etc.