Equation Formulation KCL, KVL, BCR

NA, MNA, STA

$$Ax = b$$

Solve this system of equations to obtain x for a given b

Gaussian elimination (GE)

Triangularization $\rightarrow$

$$\begin{bmatrix} \ddots & & U \\ & \ddots & \\ 0 & & \ddots \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2^{(2)} \\ \vdots \\ b_n^{(n)} \end{bmatrix} \quad \sim \frac{n^3}{3}$$

Solution is obtained by backward substitution $\sim \frac{n^2}{2}$.

What do we do with multiple 'b' vectors

LU decomposition $\sim \frac{n^3}{3}$

Forward / Back substitutions $\sim \frac{n^2}{2}$

$$Ax = b \longrightarrow L\underbrace{Ux}_{y} = b$$

$Ly = b$ where L is lower triangular

y obtained by forward substitution

$Ux = y \rightarrow x$ by back subst.

Pivoting of the matrix elements

Partial Pivoting for Small Pivots

$$\begin{bmatrix} 1.25\cdot 10^{-4} & 1.25 \\ 12.5 & 12.5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6.25 \\ 75 \end{bmatrix}$$

swap

$$\begin{bmatrix} 12.5 & 12.5 \\ 1.25\cdot 10^{-4} & 1.25 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 75 \\ 6.25 \end{bmatrix}$$

GE

$$\begin{bmatrix} 12.5 & 12.5 \\ 0 & 1.25 - 12.5\cdot 10^{-5} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 75 \\ 6.25 - 75\cdot 10^{-5} \end{bmatrix}$$

Rounded to 3 digits

$$\begin{bmatrix} 12.5 & 12.5 \\ 0 & 1.25 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 75 \\ 6.25 \end{bmatrix}$$

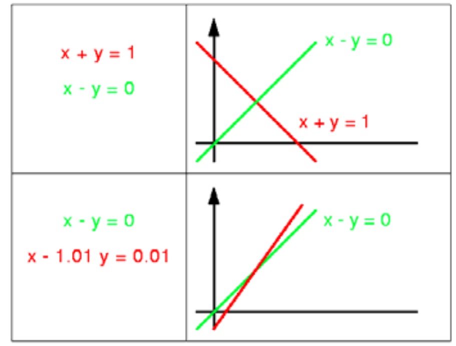
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{3digits} = \begin{bmatrix} 1 \\ 5 \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}_{5digits} = \begin{bmatrix} 1.0001 \\ 4.9999 \end{bmatrix}$$

From: A. Nardi

Error Mechanism

- ILL Conditioning (almost singular)
  Bad Luck
- Numerical Stability of Method

ILL Conditioning



From: A. Sangiovanni-Vincentelli

Error Mechanisms

- **Round-off error**
  - Pivoting helps
- **Ill conditioning (almost singular)**
  - Bad luck: property of the matrix
  - Pivoting does not help
- **Numerical Stability of Method**

From: A. Nardi

Ill-Conditioning : Norms

- **Norms useful to discuss error in numerical problems**
- **Norm** $\|\cdot\| : V \to \Re^+$

$$(1) \quad \|x\| > 0 \quad \text{if} \quad x \neq 0,\ x \in V$$

$$(2) \quad \|\alpha x\| = |\alpha| \|x\| \quad \text{if} \quad \alpha \in \Re,\ x \in V$$

$$(3) \quad \|x + y\| \leq \|x\| + \|y\| \quad \text{if} \quad x, y \in V$$

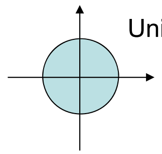
From: A. Nardi

Ill-Conditioning : Vector Norms

L$_2$ (Euclidean) norm :

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$

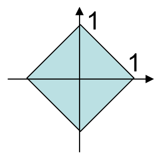
Unit circle



$$\|x\|_2 < 1$$

L$_1$ norm :

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

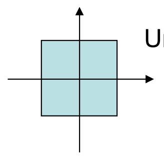


$$\|x\|_1 < 1$$

L$_\infty$ norm :

$$\|x\|_\infty = \max_i |x_i|$$

Unit square



$$\|x\|_\infty < 1$$

From: A. Nardi

Ill-Conditioning : Matrix Norms

Vector induced norm :
$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$$ = max abs column sum

$$\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$ = max abs row sum

$\|A\|_2$ = (largest eigenvalue of $A^T A$)$^{1/2}$

From: A. Nardi

Ill-Conditioning : Matrix Norms

- **More properties on the matrix norm:**

$$\|I\| = 1$$

$$\|AB\| \le \|A\| \, \|B\|$$

- **Condition Number:**

$$\kappa(A) = \|A^{-1}\| \|A\|$$

-It can be shown that: $\kappa(A) \ge 1$

-Large κ(A) means matrix is almost singular (ill-conditioned)

From: A. Nardi

Perturbation of A due to round off in GE when solving Ax=b

$$A \to A + \delta A \Rightarrow x \to x + \delta x$$

$$(A + \delta A)(x + \delta x) = b$$

$$\cancel{Ax} + A\,\delta x + \delta A x + \delta A\,\delta x = \cancel{b}$$

$$A\,\delta x + \delta A (x + \delta x) = 0$$

$$\delta x = -A^{-1} \delta A (x + \delta x)$$

$$\|\delta x\| \le \|A^{-1}\| \|\delta A\| \|x + \delta x\|$$

$$\frac{\|\delta x\|}{\|x + \delta x\|} \le \|A^{-1}\| \|A\| \frac{\|\delta A\|}{\|A\|} \quad \Rightarrow \quad \frac{\|\delta x\|}{\|x + \delta x\|} \le \kappa(A) \frac{\|\delta A\|}{\|A\|}$$

$\Rightarrow$ κ(A) large is bad

If matrix is ill-conditioned, then round-off causes problems

Perturbation in b

$$b \to b + \delta b \Rightarrow x \to x + \delta x$$

$$A(x + \delta x) = b + \delta b$$

$$\cancel{Ax} + A\delta x = \cancel{b} + \delta b$$

$$A\delta x = \delta b$$

$$\delta x = A^{-1}\delta b$$

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

$$\|\delta x\| \le \|A^{-1}\| \|\delta b\| \frac{\|b\|}{\|b\|} \le \|A^{-1}\| \frac{\|\delta b\|}{\|b\|} \|A\| \|x\|$$

$$\frac{\|\delta x\|}{\|x\|} \le \|A^{-1}\| \|A\| \frac{\|\delta b\|}{\|b\|} \quad \Rightarrow \quad \frac{\|\delta x\|}{\|x\|} \le \kappa(A) \frac{\|\delta b\|}{\|b\|}$$

$\Rightarrow$ κ(A) large is bad

From: A. Nardi

Perturbations in both A and b

$$A \to A + \delta A \text{ and } b \to b + \delta b \Rightarrow x \to x + \delta x$$

Let ε be the machine precision or resolution

Single precision: $\varepsilon \cong 10^{-8}$

Double precision: $\varepsilon \cong 10^{-16}$

For any floating point number a, $\bar{a}$ is its machine representation and $|a - \bar{a}| \le \varepsilon |a|$

$$\|\delta A\| = \|A - \bar{A}\| \le \varepsilon \|A\|$$

$$\|\delta b\| = \|b - \bar{b}\| \le \varepsilon \|b\|$$

$$\|\delta x\| = \|\delta x\|_A + \|\delta x\|_b \le 2\varepsilon\kappa(A) \|x\|$$

Numerical Stability

- **Even if the algorithm is perfect we still have an error in the solution on a computer**
- **Rounding errors may accumulate and propagate in a bad algorithm**
- **For Gaussian elimination the accumulated error is bounded**

Growth During Solution

$$\begin{bmatrix} 0.1 & & & & 1 \\ 1 & 0.1 & & & 1 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix} \xrightarrow{\text{GE Step 1}} \begin{bmatrix} 0.1 & & & & 1 \\ 0 & 0.1 & & & -9 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 & & & & 1 \\ 0 & 0.1 & & & -9 \\ & 0 & 0.1 & & 91 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix} \xrightarrow{\text{GE Step 3}} \begin{bmatrix} 0.1 & & & & 1 \\ 0 & 0.1 & & & -9 \\ & 0 & 0.1 & & 91 \\ & & 0 & 0.1 & -909 \\ & & & 1 & 0.1 \end{bmatrix}$$

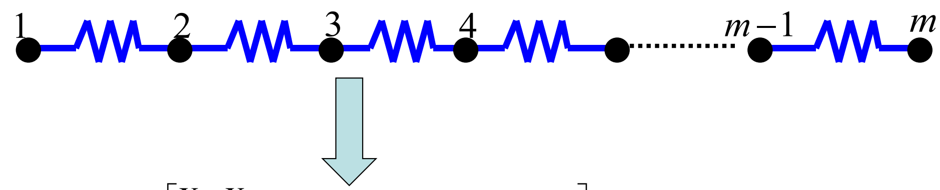
With Partial Pivoting

$$\begin{bmatrix} 0.1 & & & & 1 \\ 1 & 0.1 & & & 1 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix} \xrightarrow{\text{Reorder}} \begin{bmatrix} 1 & 0.1 & & & 1 \\ 0.1 & 0 & & & 1 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0.1 & & & 1 \\ 0 & -0.01 & & & 0.9 \\ & 1 & 0.1 & & 1 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix} \xrightarrow{\text{Reorder}} \begin{bmatrix} 0.1 & & & & 1 \\ 0 & 1 & 0.1 & & 1 \\ & -0.01 & 0 & & 0.9 \\ & & 1 & 0.1 & 1 \\ & & & 1 & 0.1 \end{bmatrix}$$

Circuit Matrices are Sparse

Example: Line of M Resistors



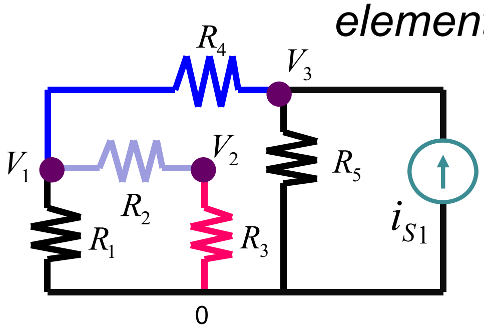
$$m \updownarrow \begin{bmatrix} X & X & & & & & \\ X & X & X & & & & \\ & X & X & X & & & \\ & & X & X & X & & \\ & & & X & X & X & \\ & & & & \ddots & \ddots & \ddots \\ & & & & X & X & X \\ & & & & & X & X & X \\ & & & & & & X & X \end{bmatrix}$$

Tridiagonal Case

From: A. Nardi

Sparse Matrices – Structural zero

Matrix element that is zero regardless of element values



Nodal Matrix

$$\begin{bmatrix} \frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_4} & -\frac{1}{R_2} & -\frac{1}{R_4} \\ -\frac{1}{R_2} & \frac{1}{R_2}+\frac{1}{R_3} & 0 \\ -\frac{1}{R_4} & 0 & \frac{1}{R_4}+\frac{1}{R_5} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ i_{s1} \end{bmatrix}$$

Symmetric
Diagonally Dominant

Structural zeros: No connection between nodes 2 and 3

From: A. Nardi

Sparse Matrices – Fill-in

Structural zero that becomes nonzero during factorization

Matrix Non zero structure

$$\begin{bmatrix} X & X & X \\ X & X & 0 \\ X & 0 & X \end{bmatrix}$$

Matrix after one LU step

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

Fill-in

From: A. Nardi

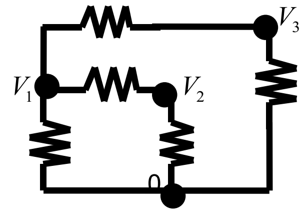
Fill-ins Propagate

$$\begin{bmatrix} X & X & X & X \\ X & X & X & X \\ 0 & X & X & X \\ 0 & X & X & X \end{bmatrix}$$

Fill-ins from Step 1 result in Fill-ins in step 2

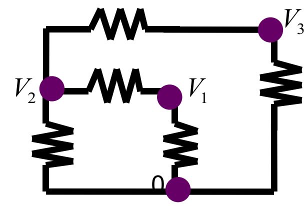
From: A. Nardi

Fill-in and Reordering



$$\Rightarrow \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$$

Fill-ins



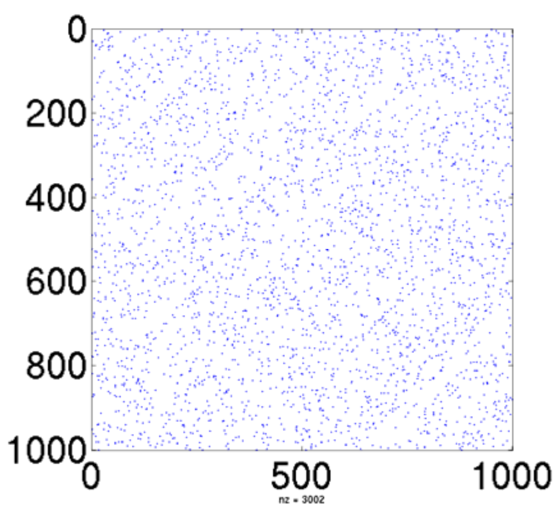
$$\Rightarrow \begin{bmatrix} x & x & 0 \\ x & x & x \\ 0 & x & x \end{bmatrix}$$

No Fill-ins

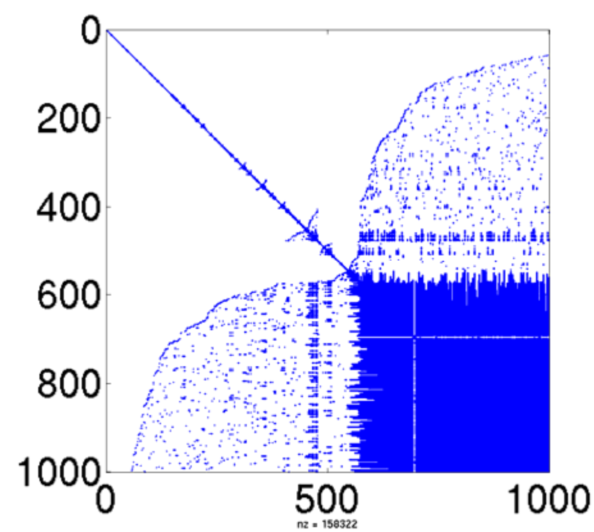
Reordering can reduce fill-in

From: A. Nardi

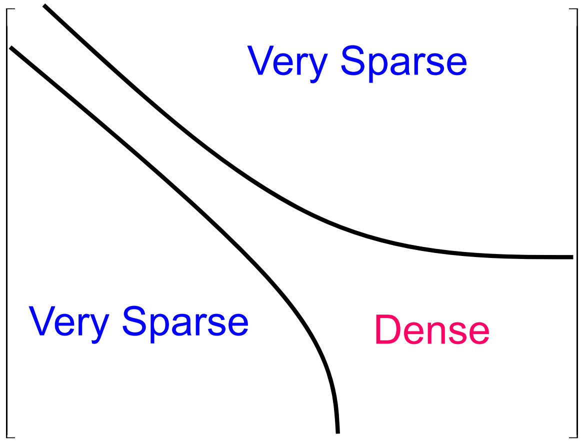
Unfactored Random Matrix



Factored Random Matrix



Pattern of a Filled-in Matrix



From: A. Nardi

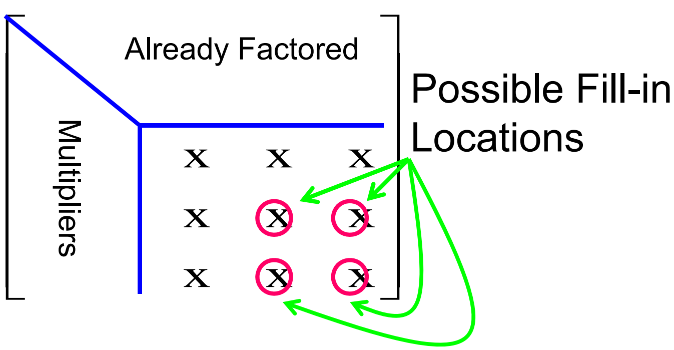
Exploiting and Maintaining Sparsity

- **Criteria for exploiting sparsity:**
  - Minimum number of ops
  - Minimum number of fill-ins
- **Pivoting to maintain sparsity: NP-complete problem $\rightarrow$ heuristics are used**
  - Markowitz, Berry, Hsieh and Ghausi, Nakhla and Singhal and Vlach
  - Choice: Markowitz
    - **Faster**
- **Pivoting for accuracy may conflict with pivoting for sparsity**

From: A. Nardi

Fill-in and Reordering

Where can fill-in occur ?



<u>Fill-in Estimate</u> = (# NZ in Row -1) (# NZ in Col -1)

Markowitz product

From: A. Nardi

Example of Fill-ins/Markowitz Reordering

$$
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}
\boxed{\begin{matrix} 4 \\ 2 \\ 2 \\ 3 \end{matrix}}
\qquad
\begin{bmatrix} 3 & 9 & 6 & 3 \\ 1 & 3 & - & - \\ - & 3 & 2 & - \\ - & 6 & 4 & 2 \end{bmatrix}
$$

$$
\boxed{\begin{matrix} 2 & 4 & 3 & 2 \end{matrix}}
$$

Markowitz products

Choose a_{21} as the pivot

Tie breaking criterion

NZUR = # of non zeros in an upper triangular row including the diagonal

NZLC = # of non zeros in a lower triangular column including the diagonal

Markowitz : 1) min (NZUR − 1) (NZLC − 1)

~1950 2) min NZLC avoid divisions for multipliers

Berry : 1) min fill-in ← actual fill in calculation

2) max NZUR

3) max NZLC

Hsieh : 1) Row Singleton (NZUR = 1)

2) Column " (NZLC = 1)

3) (NZUR, NZLC) = (2,2), (2,3), (2,4), (4,2), (3,3)

4) min (NZUR − 1) (NZLC − 1)

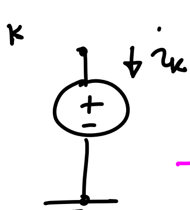
5) max NZLC

Nakhla : 1) min fill-in

2) min NZUR

3) min NZLC

MNA formulation (exploiting the matrix structure)



$\longrightarrow$

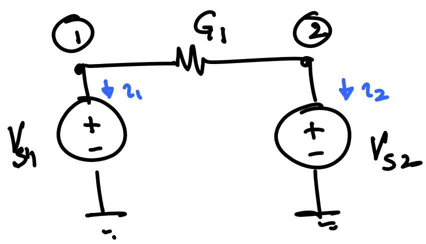
$$\begin{array}{c|cc} & k & i_k \\ \hline k & 0 & 1 \\ \text{BCR} & 1 & 0 \end{array} = [\;\;]$$

Have a singleton in a row

⇒ excellent choice for a Markowitz pivot
(Markowitz count = 0)

However, this singleton doesn't exist on a diagonal

⇒ preorder for MNA matrices



$$\begin{array}{cccc} V_1 & V_2 & i_1 & i_2 \\ \end{array}$$
$$\begin{bmatrix} G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

row singletons

Swap rows 1 & 3, 2 & 4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ G_1 & -G_1 & 1 & 0 \\ -G_1 & G_1 & 0 & 1 \end{bmatrix}$$

SPICE Pivoting Strategy

1) Choose "Markowitz element" on main diagonal a_M
2) Check with the largest element in column $|a_{max}|$

if $|a_M| < \epsilon_a + \epsilon_r |a_{max}|$ reject pivot and go back to step 1

ϵ_a: PIVTOL 10^{-13}

ϵ_r: PIVREL 10^{-3}

If all elements on diagonal fail test, select pivot outside diagonal

GMIN $= 10^{-12}$

If GMIN is reduced the PIVTOL should also be reduced

Working with sparse matrices

$n = 1000$

dense matrix 1000×1000 10^6 numbers

$\rightarrow$ 8 MB of storage

Gaussian elimination $\sim n^3$ operations

10^9 flops

If we exploit sparsity:

Store only the nonzero terms of the matrix

MNA matrices $\sim$ 3 nonzeros/row

~ 3000 nonzeros

GE $\sim n^{1.1} - n^{1.5}$ or 2k–32k flops

Need to work with sparse matrix techniques

- do not store zeros (use special data structures)
- Avoid trivial operations $0x = 0$, $1x = x$, $0 + x = x$ . . .

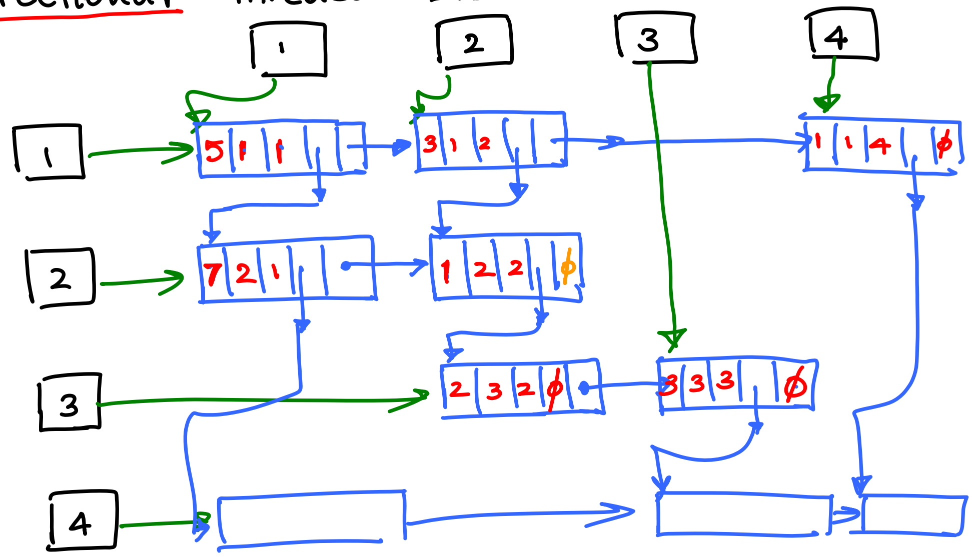
- avoid loosing sparsity

Example

$$A = \begin{bmatrix} 5 & 3 & \boxed{0} & 1 \\ 7 & 1 & \boxed{0} & \boxed{0} \\ \boxed{0} & 2 & 3 & \boxed{0} \\ -1 & \boxed{0} & 5 & 2 \end{bmatrix}$$

$$A_{ij} = \begin{bmatrix} 5 \\ 3 \\ 1 \\ 7 \\ 1 \\ 2 \\ 3 \\ -1 \\ 5 \\ 2 \end{bmatrix} \quad I = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 4 \\ 4 \\ 4 \end{bmatrix} \quad J = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 1 \\ 2 \\ 2 \\ 3 \\ 1 \\ 3 \\ 4 \end{bmatrix}$$

Bidirectional Threaded List



In C language

Element = struct { double value
int row
int col
ptr(Element) * nextInCol
ptr(Element) * nextInRow

Functions:

CreateElement (Matrix, Row, Col)
- creates and splices a new element in Matrix

get Element (Matrix, Row, Col)
- return element if it exists otherwise calls create element

CreateFillin (Matrix, Row, Col)
- creates a fill-in

ExchangeRowAndCol (Matrix, Row1, Row2, Col1, Col2)