Matrices, Graphs, and PDEs: A journey of unexpected relations

Mario Arioli[1]

[1]BMS Berlin visiting Professor
mario.arioli@gmail.com
http://www3.math.tu-berlin.de/Vorlesungen/SS14/MatricesGraphsPDEs/
Thanks Oliver

Lecture on sparse direct solvers

- Sparse matrices
- Ratherford-Boeing format - indirect addressing
- Basic operations on sparse structures:
  - permutation
  - scalar product between sparse vectors
  - linear combination of sparse vectors
- Fill-in problem in LU decomposition
- Block Triangular Form (BTF), Reverse Cuthill-MacKee

Sparse matrices

What is a SPARSE MATRIX $\mathbf{A} \in \mathbf{R}^{n \times n}$?

Sparse matrices

What is a SPARSE MATRIX $\mathbf{A} \in \mathbf{R}^{n \times n}$?

- The number of non zero entries in $\mathbf{A}$ ($nz(\mathbf{A})$) must be $\mathcal{O}(n)$ i.e. If we define $\delta \in (0, 1)$ the density of non zeros entries $nz(\mathbf{A}) = \delta n^2$, we assume that

$$\frac{1}{n} < \delta = \mathcal{O}(\frac{1}{n})$$

Sparse matrices

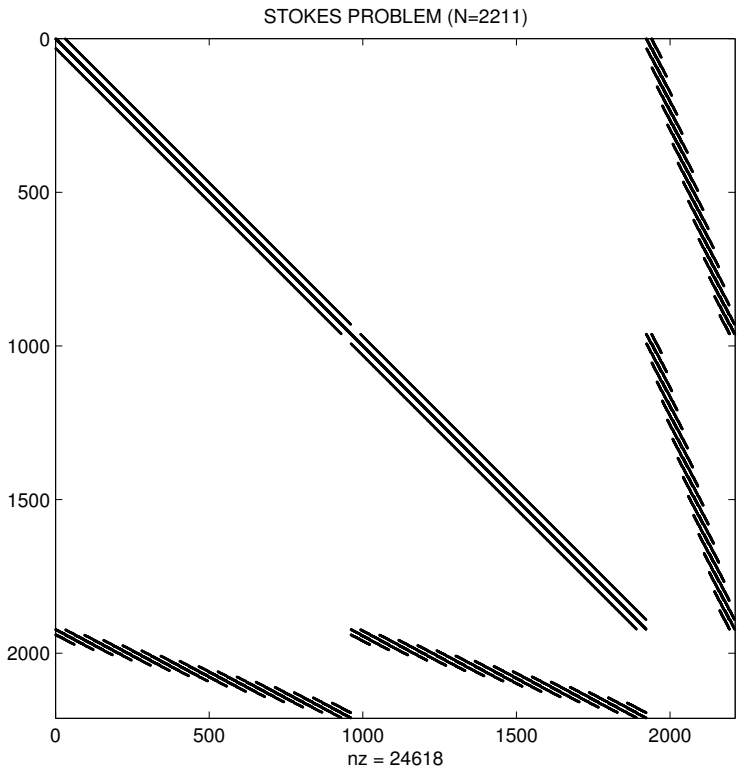
What is a SPARSE MATRIX $\mathbf{A} \in \mathrm{R}^{n \times n}$?

- The number of non zero entries in $\mathbf{A}$ ($nz(\mathbf{A})$) must be $\mathcal{O}(n)$ i.e. If we define $\delta \in (0, 1)$ the density of non zeros entries $nz(\mathbf{A}) = \delta n^2$, we assume that

$$\frac{1}{n} < \delta = \mathcal{O}(\frac{1}{n})$$

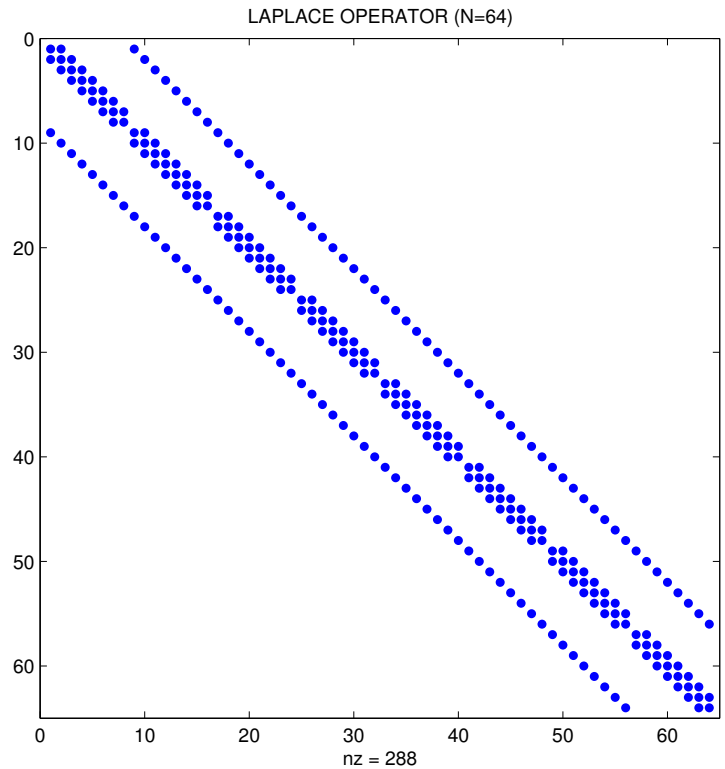
- The previous statement implies that $\mathbf{A}$ depends on a "small" number of parameters, but the reverse is not true (Vandermonde, circulant matrices, etc....)

Sparse matrices: examples



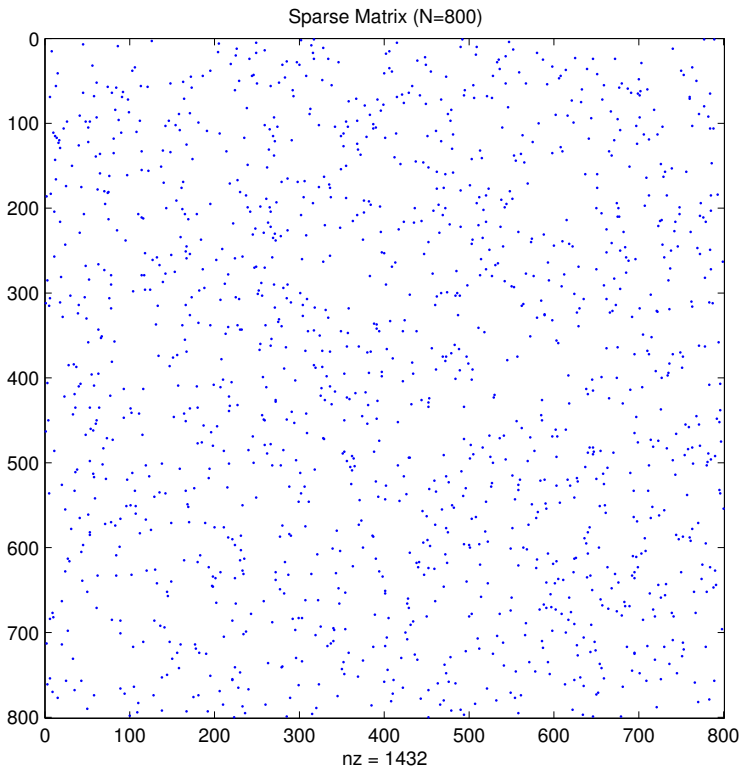
Stokes problem approximation by mixed finite element method
($n = 2211$)

Sparse matrices: examples



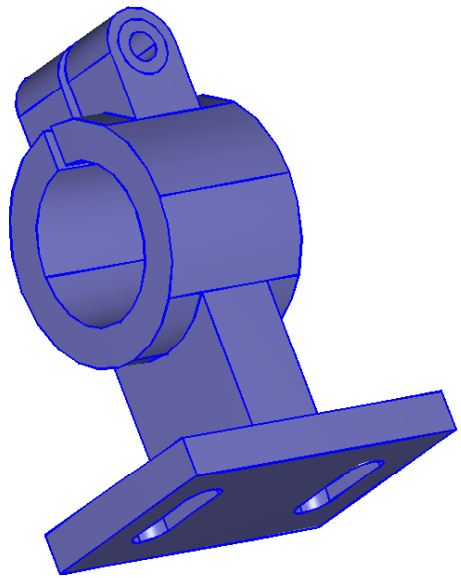
Laplace operator approximation by the finite difference method ($n = 64$)

Sparse matrices: examples



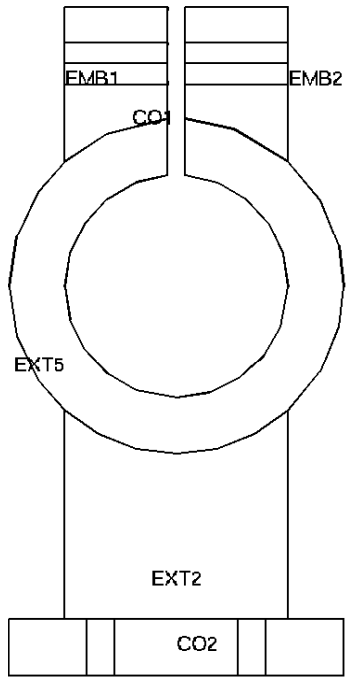
General non symmetric sparse matrix ($n = 800$)

Sparse matrices: a more complex example



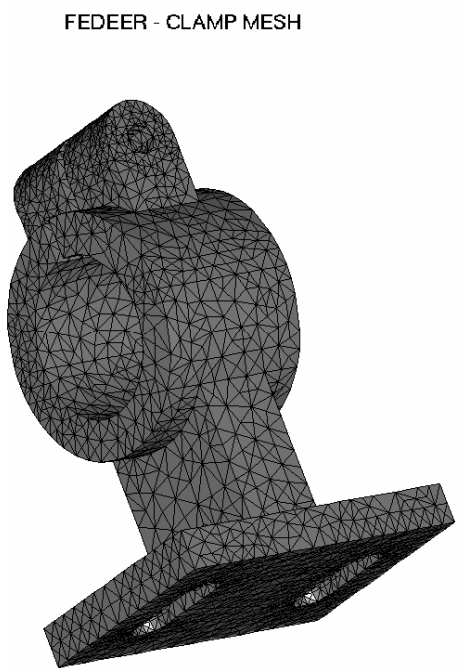
Structural mechanics problem

Sparse matrices: a more complex example



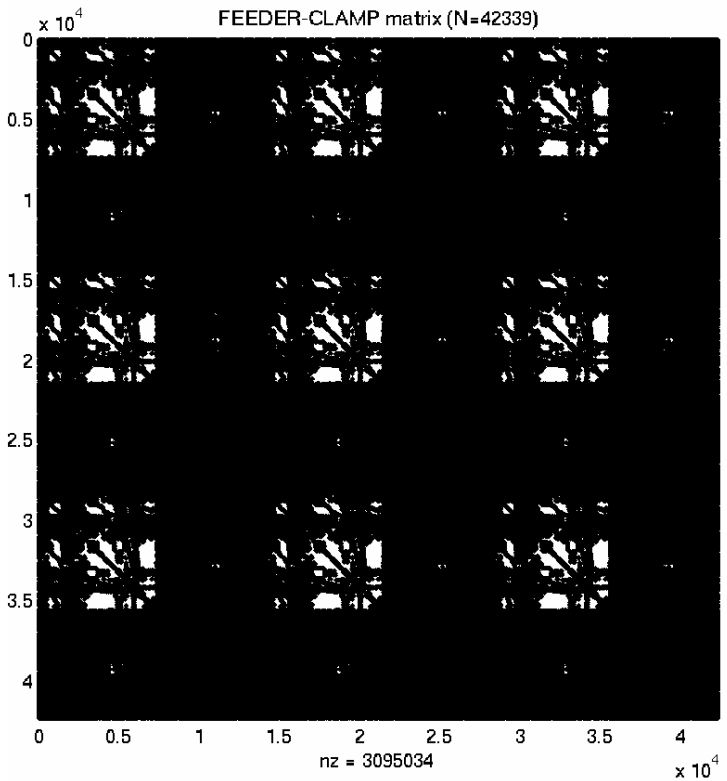
Structural mechanics problem

Sparse matrices: a more complex example



Mesh

Sparse matrices: a more complex example



A with $n = 42339$ $nz(\mathbf{A}) = 3095034$ and $\delta = 0.0017$

Sparse matrices: Ambiguous definition

- What is a good δ?

Sparse matrices: Ambiguous definition

- What is a good δ?
- If $\mathbf{A}$ is a vector: $\delta = 0.1$? $\delta = 0.001$?

Sparse matrices: Ambiguous definition

- What is a good δ?
- If $\mathbf{A}$ is a vector: $\delta = 0.1$? $\delta = 0.001$?
- The choice will depend on n

Sparse vectors: storage

A sparse vector may be held in a full-lenght vector storage: rapid access but arather wasteful of storage.

Sparse vectors: storage

A sparse vector may be held in a full-lenght vector storage: rapid access but arather wasteful of storage.

- If $\mathbf{v}$ is a n-vector: (v_i, i) is enough. We have a real vector `VAL` and an integer vector `IND`, each of lenght at least the number of entries (`NZV` $\leq$ `N`. `VAL(J)`$= v_i$ and `IND(J)` $= i$ `J = 1,NZV`.

Sparse vectors: storage

A sparse vector may be held in a full-lenght vector storage: rapid access but arather wasteful of storage.

- If $\mathbf{v}$ is a n-vector: (v_i, i) is enough. We have a real vector `VAL` and an integer vector `IND`, each of lenght at least the number of entries (`NZV` $\leq$ `N`. `VAL(J)`$= v_i$ and `IND(J)` $= i$ `J = 1,NZV`.
- **gather** is the operation of transforming a full vector to packed form
- **scatter** is the reverse operation

Sparse vectors: inner product of two packed vectors

Let (VALX,INDX) and (VALY,INDY) two packed vectors of length NZX and NZY

```
do K = 1,N
      W(K) = 0
end do
PROD = 0
do KX = 1,NZX
      W(INDX(KX)) = VALX(KX)
end do
do KY = 1,NZY
      PROD = PROD + W(INDY(KY))*VAL(KY)
end do
```

Sparse vectors: $\mathbf{x} = \mathbf{x} + \alpha\mathbf{y}$

Let (VALX,INDX) and (VALY,INDY) two packed vectors of lenght NZX and NZY

```
do K = 1,NZY
          W(INDX(K)) = VALY(K)
end do
do KX = 1,NZX
          I = INDX(KX)
          if (W(I) .NE. ZERO) then
                    VALX(KX) = VALX(KX) + ALPHA*W(I)
                    W(I) = ZERO
          end if
end do
do KY = 1,NZY
          I = INDX(KY)
          if (W(I) .NE. ZERO) then
                    NZX = NZX + 1
                    VALX(NZX) = ALPHA*W(I)
                    INDX(NZX) = I
                    W(I) = ZERO
          end if
end do
```

Sparse vectors: $\mathbf{x} = \mathbf{x} + \sum_j \alpha_j \mathbf{y}^{(j)}$

1. Load $\mathbf{x}$ in $\mathbf{w}$
2. For each $\mathbf{y}^{(j)}$: Scan $\mathbf{y}^{(j)}$. For each $y_i^{(j)}$, check w_i. If is nonzero, set $w_i = w_i + \alpha_j y_i^{(j)}$ and add i to the data structure for $\mathbf{x}$
3. Scan the revised data structure for $\mathbf{x}$. For each i, set $x_i = w_i,\ w_i = 0$

Sparse vectors: permutations

A permutation matrix $\mathbf{P}$ can be represented by a set of integers

$$ipos(i) \qquad i = 1, \ldots, n$$

which represent the position of the components of $\mathbf{x}$ in $\mathbf{y} = \mathbf{P}\mathbf{x}$

The permuted vector can be computed as follow

```
do I = 1,N
    Y(IPOS(I)) = X(I)
end do
```

The inverse permutation *invipos* can be easily computed

```
do I = 1,N
    INVIPOS(IPOS(I)) = I
end do
```

Sparse vectors: permutations

A permutation matrix $\mathbf{P}$ can be represented by a set of integers

$$ipos(i) \qquad i = 1, \ldots, n$$

which represent the position of the components of $\mathbf{x}$ in $\mathbf{y} = \mathbf{P}\mathbf{x}$
example: $ipos = [2, 4, 3, 1, 6, 5, 8, 7]$
The permuted vector can be computed as follow

```
do I = 1,N
    Y(IPOS(I)) = X(I)
end do
```

The inverse permutation *invipos* can be easily computed

```
do I = 1,N
    INVIPOS(IPOS(I)) = I
end do
```

Sparse vectors: permutations

A permutation matrix $\mathbf{P}$ can be represented by a set of integers

$$ipos(i) \qquad i = 1, \ldots, n$$

which represent the position of the components of $\mathbf{x}$ in $\mathbf{y} = \mathbf{P}\mathbf{x}$

example: $ipos = [2, 4, 3, 1, 6, 5, 8, 7]$

The permuted vector can be computed as follow

```
do I = 1,N
    Y(IPOS(I)) = X(I)
end do
```

example: $\mathbf{x} = [x_1, 0, 0, x_4, 0, 0, x_7, 0]$, $\mathbf{y} = [0, x_4, 0, x_1, 0, 0, 0, x_7]$

The inverse permutation *invipos* can be easily computed

```
do I = 1,N
    INVIPOS(IPOS(I)) = I
end do
```

Sparse vectors: permutations

A permutation matrix $\mathbf{P}$ can be represented by a set of integers

$$ipos(i) \qquad i = 1, \ldots, n$$

which represent the position of the components of $\mathbf{x}$ in $\mathbf{y} = \mathbf{P}\mathbf{x}$

example: $ipos = [2, 4, 3, 1, 6, 5, 8, 7]$

The permuted vector can be computed as follow

```
do I = 1,N
    Y(IPOS(I)) = X(I)
end do
```

example: $\mathbf{x} = [x_1, 0, 0, x_4, 0, 0, x_7, 0]$, $\mathbf{y} = [0, x_4, 0, x_1, 0, 0, 0, x_7]$

The inverse permutation *invipos* can be easily computed

```
do I = 1,N
    INVIPOS(IPOS(I)) = I
end do
```

example: $invipos = [4, 1, 3, 2, 6, 5, 8, 7]$

Sparse vectors: permutations

The sparse vector **x** can be permuted in place by

```
do I = 1,NZX
    INDX(I) = INVIPOS(INDX(I))
end do
```

example: $\mathbf{x} = [x_1, 0, 0, x_4, 0, 0, x_7, 0]$, $\mathbf{y} = [0, x_4, 0, x_1, 0, 0, 0, x_7]$

$invipos = [4, 1, 3, 2, 6, 5, 8, 7]$

$indx = [1, 4, 7] \quad \rightarrow indx = [4, 2, 8]$

Sparse matrices: storage

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{bmatrix}$$

Coordinate scheme

Subscripts	1	2	3	4	5	6	7	8	9	10	11
IRN	1	2	2	1	5	3	4	5	2	4	5
JCN	4	5	1	1	5	2	4	3	3	2	1
VAL	-1	3	2	1	6	-3	-4	-5	-2	4	5

Sparse matrices: storage

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 2 & 0 & -2 & 0 & 3 \\ 0 & -3 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{bmatrix}$$

Collection of sparse row vectors

Subscripts	1	2	3	4	5	6	7	8	9	10	11
LENROW	2	3	1	2	3						
IROWST	1	3	6	7	9						
JCN	4	1	5	1	3	2	4	2	3	1	5
VAL	-1	1	3	2	-2	-3	-4	4	-5	5	6

Sparse matrices: fill-in and reordering

$$\mathbf{A} = \begin{bmatrix} x & x & x & x & x & x & x & x \\ x & x & & & & & & \\ x & & x & & & & & \\ x & & & x & & & & \\ x & & & & x & & & \\ x & & & & & x & & \\ x & & & & & & x & \\ x & & & & & & & x \end{bmatrix}$$

Sparse matrices: fill-in and reordering

$$\mathbf{L} = \begin{bmatrix} x & & & & & & & \\ x & x & & & & & & \\ x & x & x & & & & & \\ x & x & x & x & & & & \\ x & x & x & x & x & & & \\ x & x & x & x & x & x & & \\ x & x & x & x & x & x & x & \\ x & x & x & x & x & x & x & x \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} x & x & x & x & x & x & x & x \\ & x & x & x & x & x & x & x \\ & & x & x & x & x & x & x \\ & & & x & x & x & x & x \\ & & & & x & x & x & x \\ & & & & & x & x & x \\ & & & & & & x & x \\ & & & & & & & x \end{bmatrix}$$

Sparse matrices: fill-in and reordering

$$\mathbf{P}^T\mathbf{A}\mathbf{P} = \begin{bmatrix} x & & & & & & & x \\ & x & & & & & & x \\ & & x & & & & & x \\ & & & x & & & & x \\ & & & & x & & & x \\ & & & & & x & & x \\ & & & & & & x & x \\ x & x & x & x & x & x & x & x \end{bmatrix}$$

Sparse matrices: fill-in and reordering

$$\mathbf{L} = \begin{bmatrix} x & & & & & & & \\ & x & & & & & & \\ & & x & & & & & \\ & & & x & & & & \\ & & & & x & & & \\ & & & & & x & & \\ & & & & & & x & \\ x & x & x & x & x & x & x & x \end{bmatrix} \qquad \mathbf{U} = \begin{bmatrix} x & & & & & & & x \\ & x & & & & & & x \\ & & x & & & & & x \\ & & & x & & & & x \\ & & & & x & & & x \\ & & & & & x & & x \\ & & & & & & x & x \\ & & & & & & & x \end{bmatrix}$$

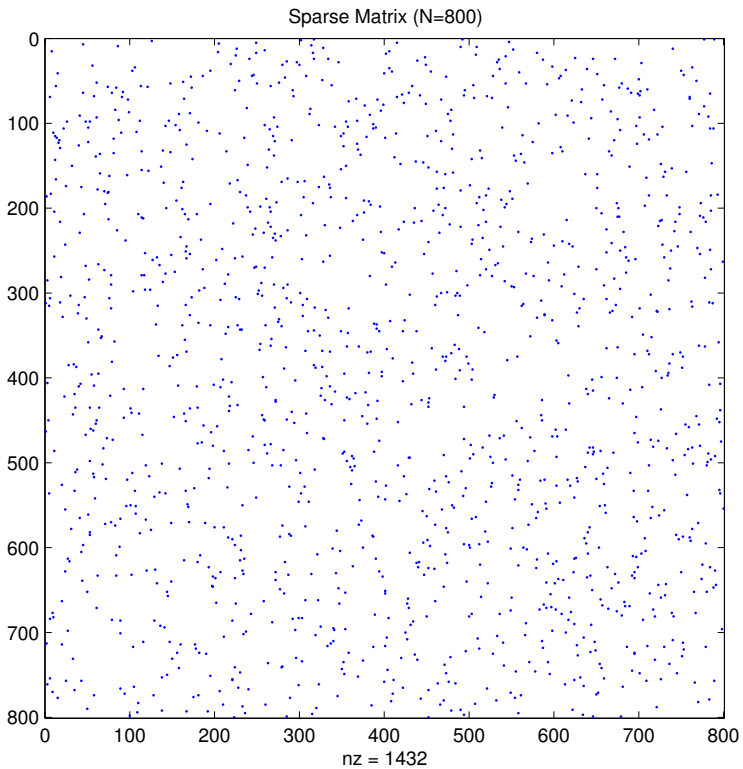
Sparse matrices: Block Triangular Form (BTF)

We seek permutations $\mathbf{P}$ and $\mathbf{Q}$ such that $\mathbf{A}$ is in block triangula form.

$$\mathbf{PAQ} = \begin{bmatrix} \mathbf{B}_{11} & & & \\ \mathbf{B}_{21} & \mathbf{B}_{22} & & \\ \vdots & \vdots & \vdots & \\ \mathbf{B}_{k1} & \dots & \mathbf{B}_{k,k-1} & \mathbf{B}_{kk} \end{bmatrix}$$

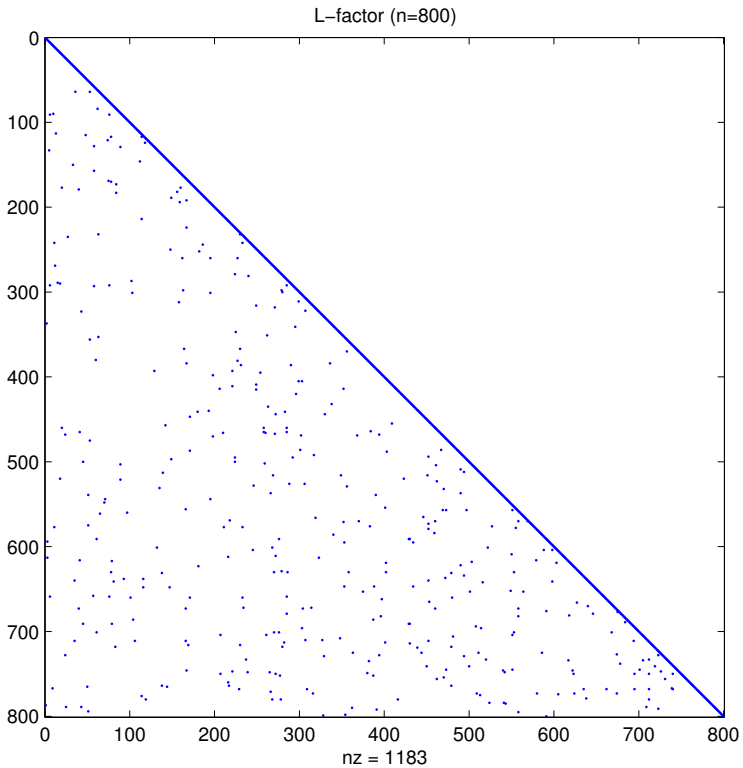
If the matrix $\mathbf{A}$ can be permute in BTF, we call it reducible, otherwise $\mathbf{A}$ is irriducible. We assume that all blocks $\mathbf{B}_{ii}$ are irriducibles.

Block Triangular Form (BTF): example



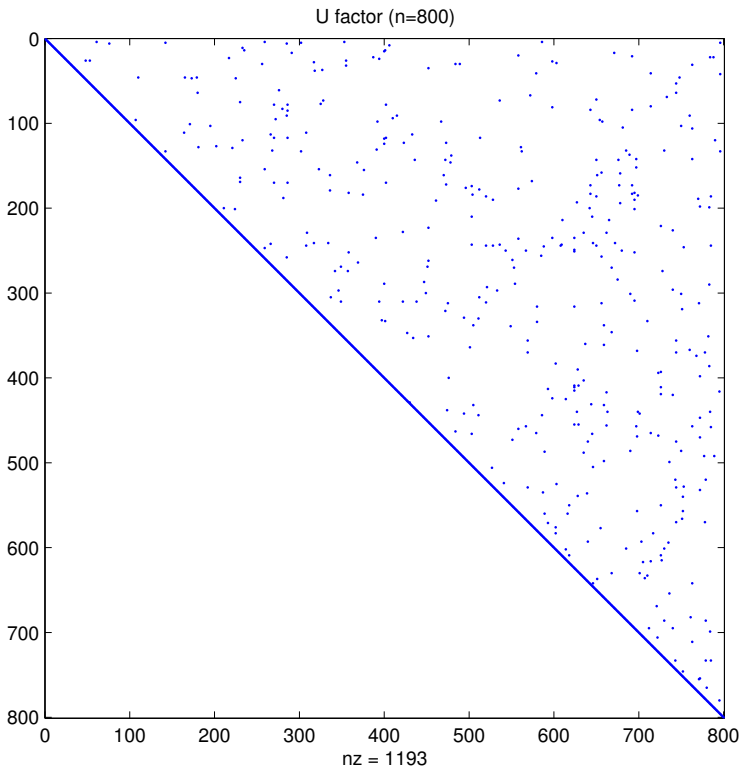
General non symmetric sparse matrix ($n = 800$)

Block Triangular Form (BTF): example



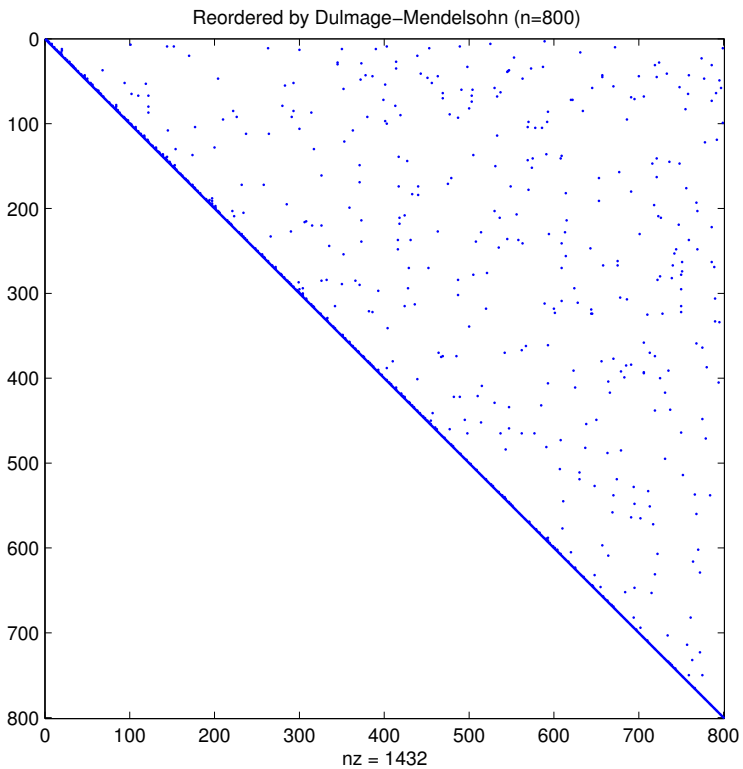
L factor ($n = 800$)

Block Triangular Form (BTF): example



U factor ($n = 800$)

Block Triangular Form (BTF): example



After application of Dulmage-Mendelsohn algorithm in Matlab ($n = 800$)

In Matlab it is better to compute the upper triangular version

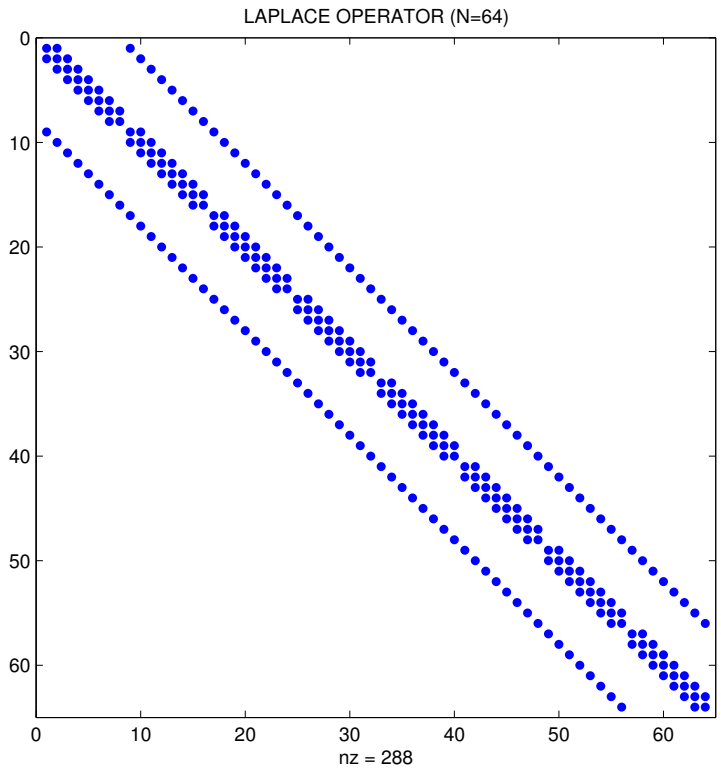
Block Triangular Form (BTF)

The algorithm (see Duff, Erisman, and Reid Ch-6) is based of two separate phases

- Maximum Transversal by computing $\mathbf{P}$: the diagonal of $\mathbf{PA}$ is full.
- Symmetric permutation of $\mathbf{PA}$ to Lower BTF: $\mathbf{Q}^T(\mathbf{PA})\mathbf{Q}$ Tarjan (1972), Duff and Reid (1978) , HSL MC23 FORTRAN implementation (75 instructions)

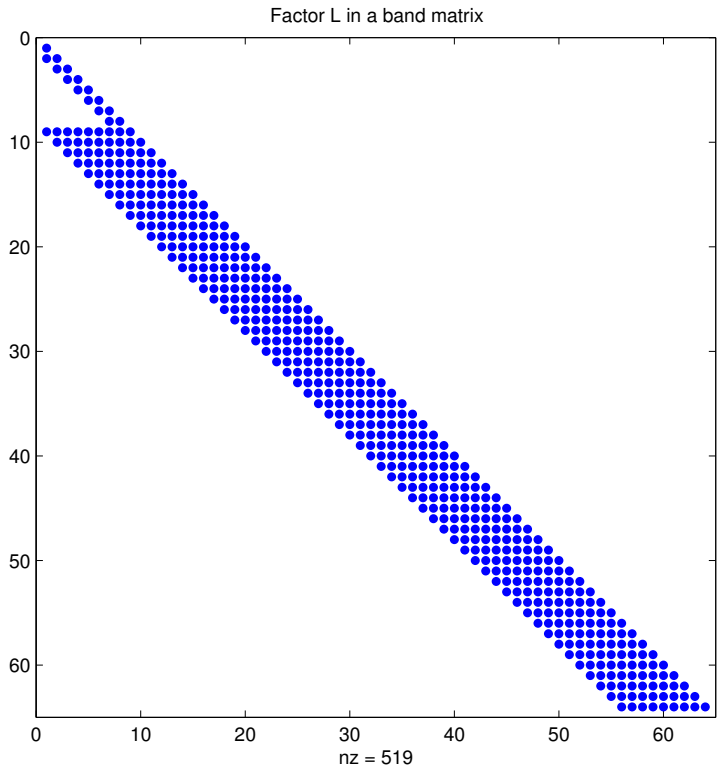
Block Tridiagonal Form and graphs

A symmetrically structured matrix has bandwidth $2m-1$ and semibandwidth m if m is the smallest integer such that $a_{ij}=0$ whenever $|i-j|>m$. The fill-in is confined in the band.



Laplace operator approximation by the finite difference method ($n=64$, $m=8$)

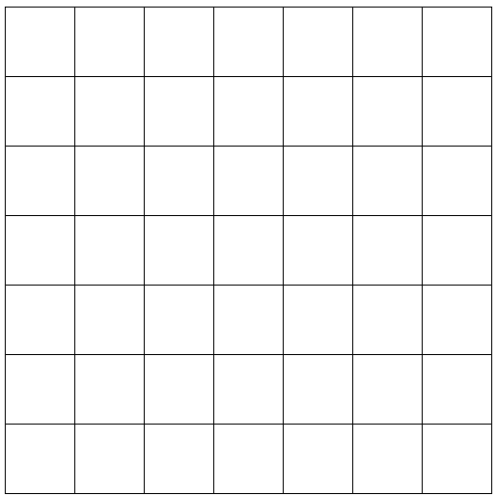
Block Tridiagonal Form and graphs



Laplace operator approximation by the finite difference method ($n = 64$, $m = 8$) L factor

Block Tridiagonal Form and graphs

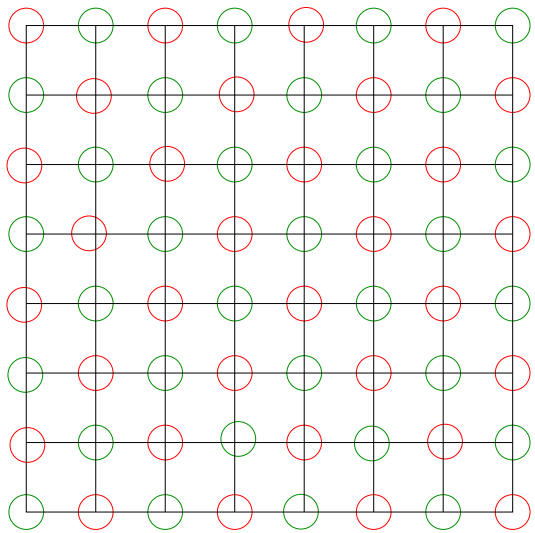
We can associate to the symmetrically structured matrix a graph and any renumbering of the nodes corresponds to a symmetric permutation of the matrix. The number of arcs incident in one node is the degree of the node. The degree of node i corresponds to the number of nonzeros in row i of the matrix.



Laplace operator approximation by the finite difference method ($n = 64$, $m = 8$)

Block Tridiagonal Form and graphs

We divide the nodes in **level sets** S_i. S_1 consists of one node of minimum degree. The general level set S_l consists of all the neighbours of nodes in S_{i-1} that are not in S_{i-1} or S_{i-2}.



Laplace operator approximation by the finite difference method ($n = 64$, $m = 8$)

Block Tridiagonal Form and graphs

Renumbering the nodes in the level sets, we have a tridiagonal matrix where the diagonal block i consists of the nodes in level set S_i.
We can improve the number of diagonal block restarting the process from one of the nodes in the final level set (pseudoperipheral nodes).
The resulting order is called the Cuthill-McKee order.
George (1971) proved that reversing the order (Reverse Cuthill-McKee order) decreases the fill-in in the Gauss factorization.

Ordering, Frontal, Multifrontal

- Ordering of sparse matrices: Nested Dissection, minimum degree, and approximate minimum degree.
- Threshold pivoting and its control
- Frontal e multifrontal methods

Cholesky via Schur complement

$$
\begin{aligned}
\mathbf{A} &= \begin{bmatrix} d_1 & \mathbf{v}_1^T \\ \mathbf{v}_1 & \mathbf{H}_1 \end{bmatrix} \\
&= \begin{bmatrix} \sqrt{d_1} & 0 \\ \mathbf{v}_1/\sqrt{d_1} & \mathbf{I}_{n-1} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{S_1} \end{bmatrix} \begin{bmatrix} \sqrt{d_1} & \mathbf{v}_1^T/\sqrt{d_1} \\ 0 & \mathbf{I}_{n-1} \end{bmatrix} \\
&= \mathbf{L}_1 \mathbf{A}_1 \mathbf{L}_1^T \\
\mathbf{S}_1 &= \mathbf{H}_1 - \frac{\mathbf{v}_1 \mathbf{v}_1^T}{d_1}
\end{aligned}
$$

Repeat the operation on $\mathbf{S}_1$. $\mathbf{L} = \mathbf{L}_{n-1} \ldots \mathbf{L}_1$

Cholesky via Graph theory: notations

$$\begin{aligned} \mathcal{P}(\mathbf{A}) &= \{(i,j)|a_{ij} \neq 0 \text{ and } i \neq j\} \\ \mathbf{F} &= \mathbf{L} + \mathbf{L}^T \\ \mathcal{P}(\mathbf{F}) &= \{(i,j)|f_{ij} \neq 0 \text{ and } i \neq j\} \end{aligned}$$

$\mathcal{P}(\mathbf{B})$ is the pattern of the matrix $\mathbf{B}$. Let $(\mathcal{N})$ be the set $\{1,\ldots,n\}$ and $\mathcal{G}_{\mathbf{B}} = (\mathcal{N}, \mathcal{P}(\mathbf{B}))$ the graph with nodes $(\mathcal{N})$ and undirected arcs $\mathcal{P}(\mathbf{B})$. We denote by $Adj(i) = \{j|j \neq i, (i,j) \in \mathcal{P}(\mathbf{B})\}$

$$\mathcal{P}(\mathbf{A}) \subseteq \mathcal{P}(\mathbf{F})$$

$$Fill(\mathbf{A}) = \mathcal{P}(\mathbf{F}) - \mathcal{P}(\mathbf{A})$$

Cholesky via Graph theory: fill-in

$(\mathbf{S_1})_{\mathbf{ij}} \neq \mathbf{0}$ iff $(\mathbf{H}_1)_{ij} \neq 0$ or *both* $(\mathbf{v}_1)_i \neq 0$ and $(\mathbf{v}_1)_j \neq 0$

Cholesky via Graph theory: fill-in

$(\mathbf{S_1})_{\mathbf{ij}} \neq \mathbf{0}$ iff $(\mathbf{H}_1)_{ij} \neq 0$ or *both* $(\mathbf{v}_1)_i \neq 0$ and $(\mathbf{v}_1)_j \neq 0$

1. Delete node node x_1 and all its incident arcs
2. Add arcs to the graph so that nodes in $adj(x_1)$ are pairwise adjacent in $\mathcal{G}_{\mathbf{S_1}}$

Cholesky via Graph theory: fill-in

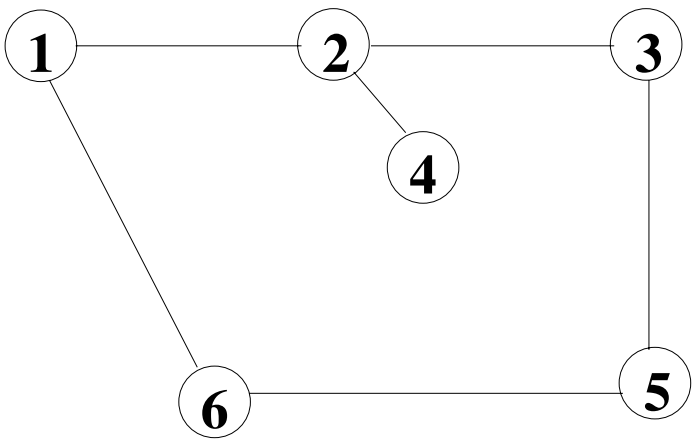
$(\mathbf{S_1})_{\mathbf{ij}} \neq \mathbf{0}$ iff $(\mathbf{H}_1)_{ij} \neq 0$ or *both* $(\mathbf{v}_1)_i \neq 0$ and $(\mathbf{v}_1)_j \neq 0$

1. Delete node node x_1 and all its incident arcs
2. Add arcs to the graph so that nodes in $adj(x_1)$ are pairwise adjacent in $\mathcal{G}_{\mathbf{S_1}}$

Lemma 1 The unordered pair $(x_i, x_j) \in \mathcal{P}_{\mathbf{F}}$ iff $(x_i, x_j) \in \mathcal{P}_{\mathbf{A}}$ or $(x_i, x_k) \in \mathcal{P}_{\mathbf{F}}$ and $(x_j, x_k) \in \mathcal{P}_{\mathbf{F}}$ for some $k < \min\{i, j\}$

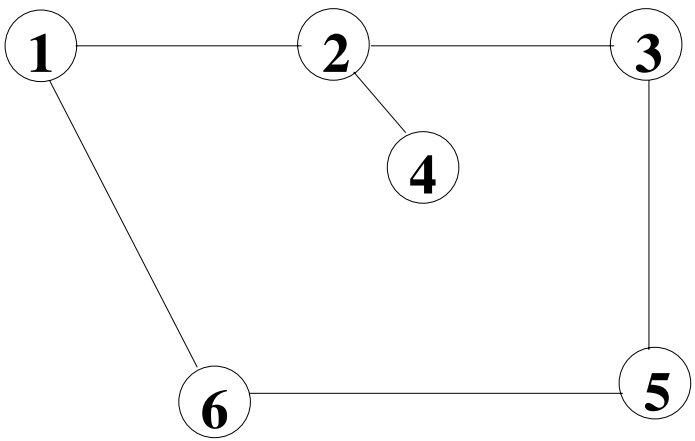
Graph theory: example

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



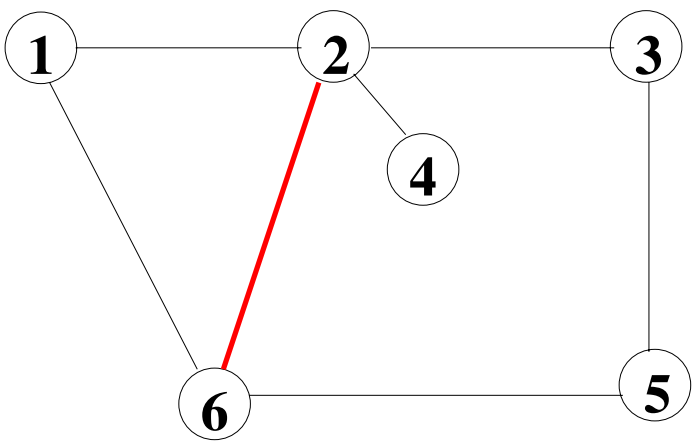
Graph theory: example

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



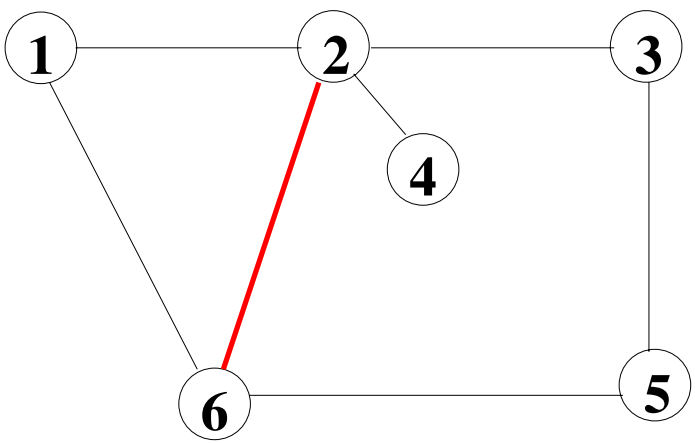
Graph theory: example

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



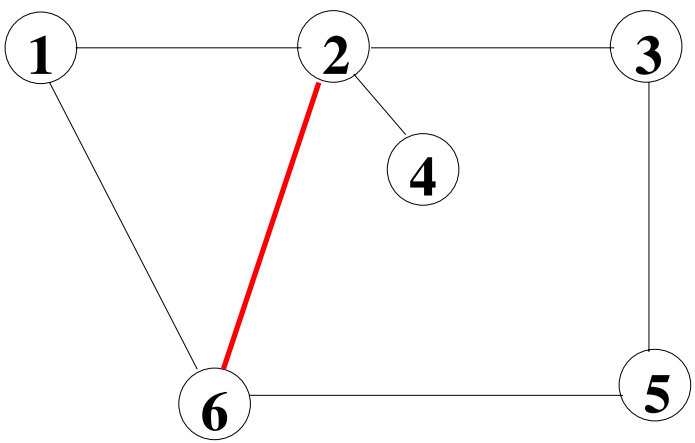
Graph theory: example

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



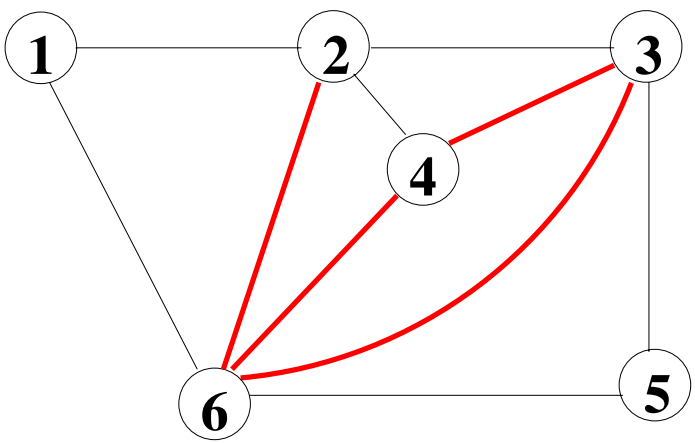
Graph theory: example

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



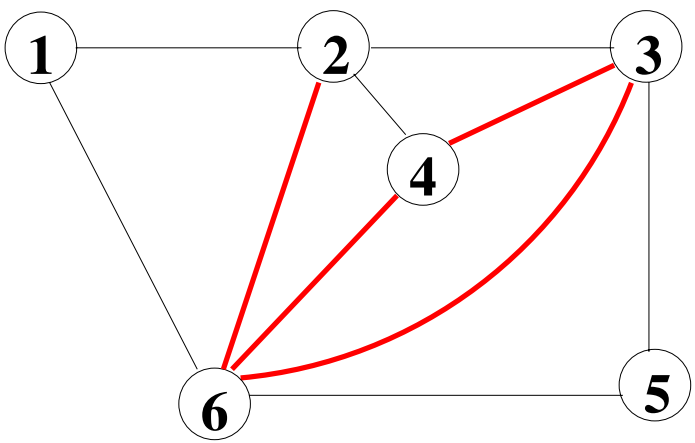
Graph theory: example

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



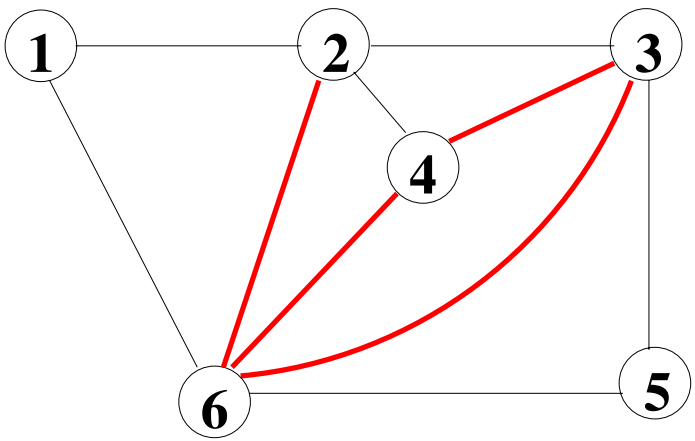
Graph theory: example

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



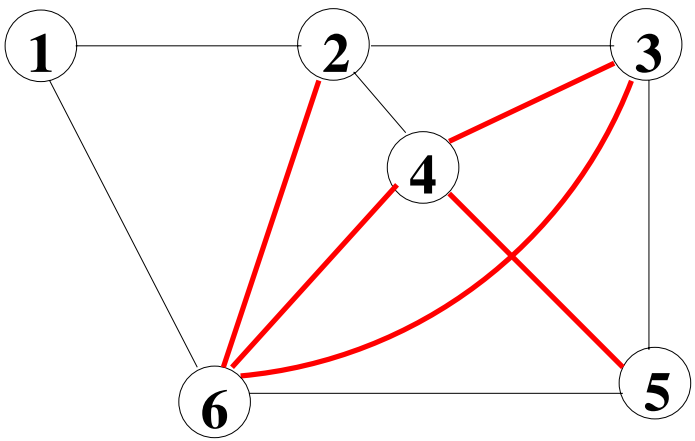
Graph theory: example

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



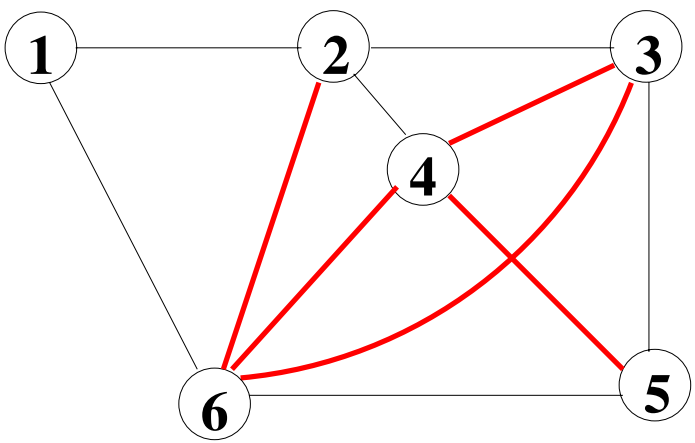
Graph theory: example

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



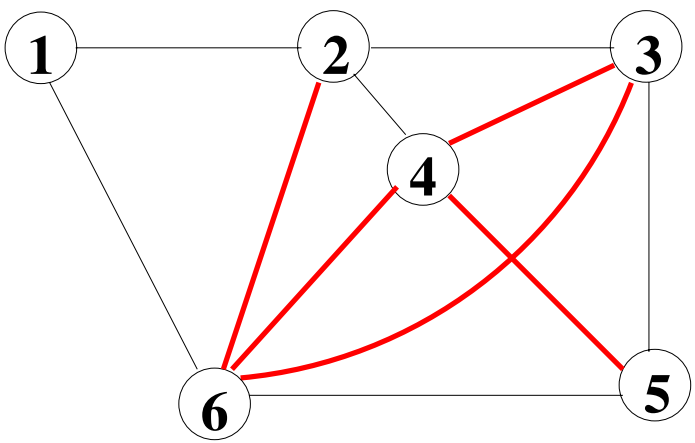
Graph theory: example

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

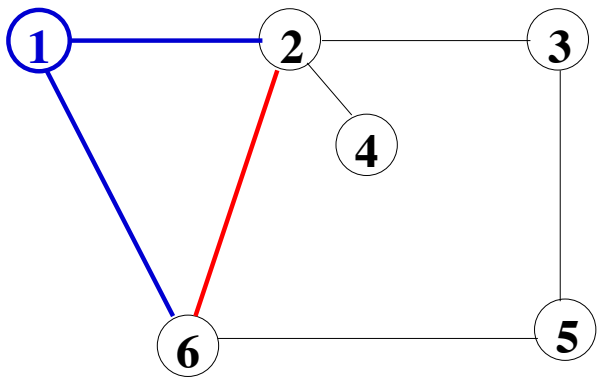


Graph theory: example

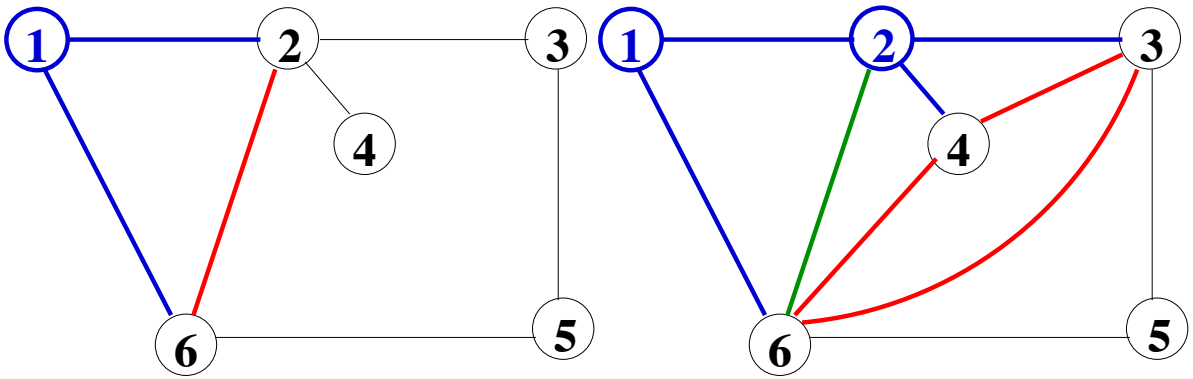
$$\begin{bmatrix} * & * & & & & * \\ & * & * & * & & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \\ & & & & & * \end{bmatrix}$$



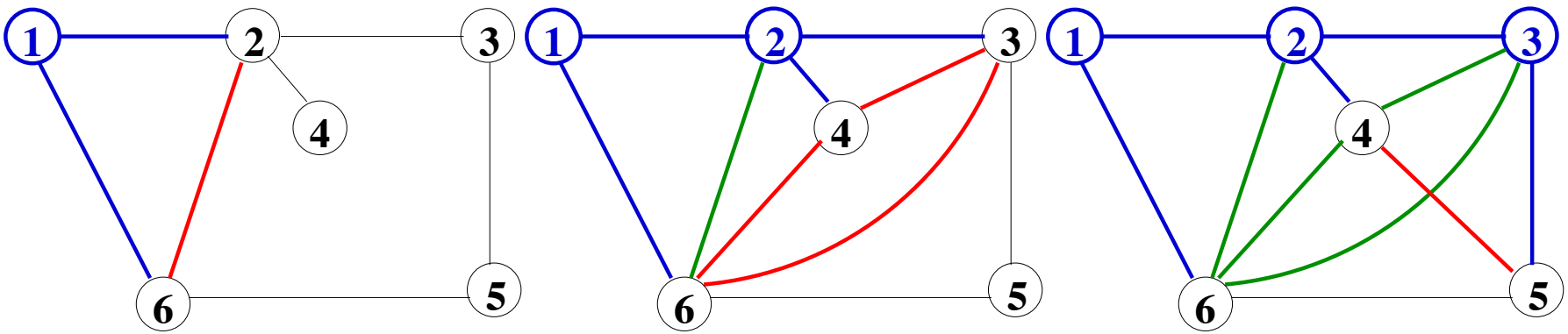
Graph theory: reachable points



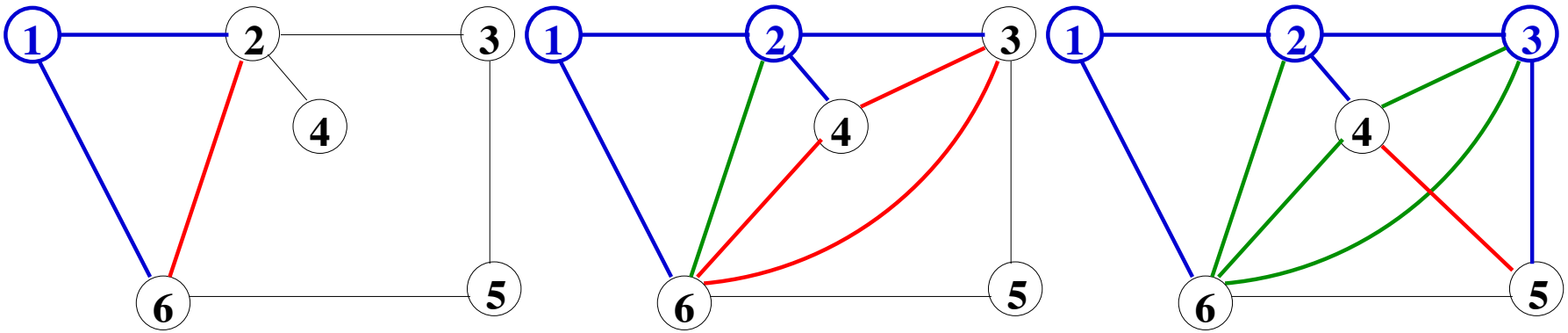
Graph theory: reachable points



Graph theory: reachable points



Graph theory: reachable points



Let $S \subset \mathcal{N}$ and $x \in \{\mathcal{N} - S\}$. x is reachable from y through S if $\exists(y, v_1, \ldots v_k, x)$ a path from x to y such that $v_i \in S$ for $1 \leq i \leq k$ ($k = 0 \Rightarrow$ any adjacent node of y not in S is reachable.

Graph theory: reachable sets

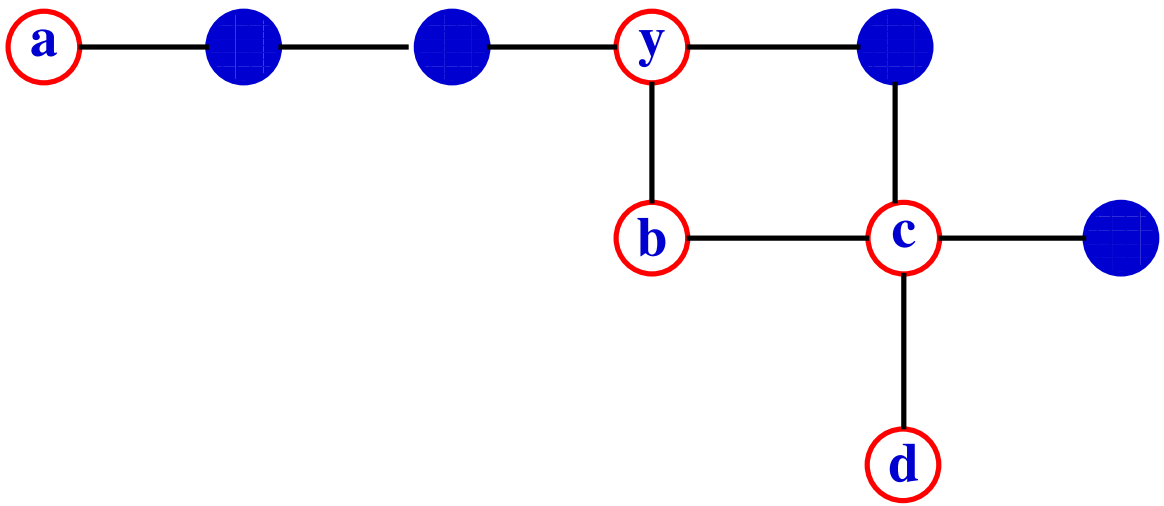
Definition Let $S \subset \mathcal{N}$. The reachable set of y through S is

$$Reach(y, S) = \{x \in \{\mathcal{N} - S\} | x \text{ is reachable from } y \text{ through } S\}$$

Graph theory: reachable sets

Definition Let $S \subset \mathcal{N}$. The reachable set of y through S is

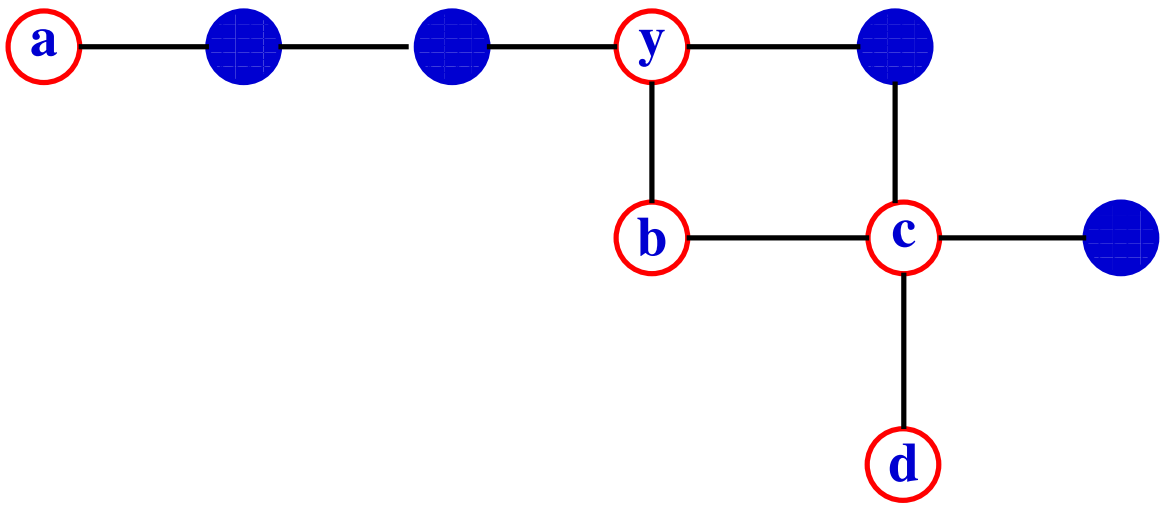
$$Reach(y, S) = \{x \in \{\mathcal{N} - S\} | x \text{ is reachable from } y \text{ through } S\}$$



Graph theory: reachable sets

Definition Let $S \subset \mathcal{N}$. The reachable set of y through S is

$$Reach(y, S) = \{x \in \{\mathcal{N} - S\} | x \text{ is reachable from } y \text{ through } S\}$$



$$Reach(y, S) = \{a, b, c\}$$

Graph theory: reachable sets

Theorem 2

$$\mathcal{G}_{\mathbf{F}} = \{\{x_i, x_j\} | x_j \in \mathit{Reach}(x_i, \{x_1, x_2, \ldots, x_{i-1}\})\}$$

Graph theory: reachable sets

Theorem 2

$$\mathcal{G}_{\mathbf{F}} = \{\{x_i, x_j\} | x_j \in \mathit{Reach}(x_i, \{x_1, x_2, \ldots, x_{i-1}\})\}$$

Let $\mathcal{G}_i$ be the graph of $\mathbf{S_i}$ the i-th Schur complement

Theorem 3

Let y be a node in $\mathcal{G}_i$. The set of nodes adjacent to y in $\mathcal{G}_i$ is given by

$$\mathit{Reach}(y, \{x_1, \ldots, x_{i-1}\})$$

Minimum degree algorithm

1. (*Initialization*) $S \leftarrow \emptyset$. $Deg(x) \leftarrow |Adj(x)| \; \forall x \in \mathcal{N}$
2. (*Minimum degree selection*) Pick a node $y \in \{\mathcal{N} - S\}$ where

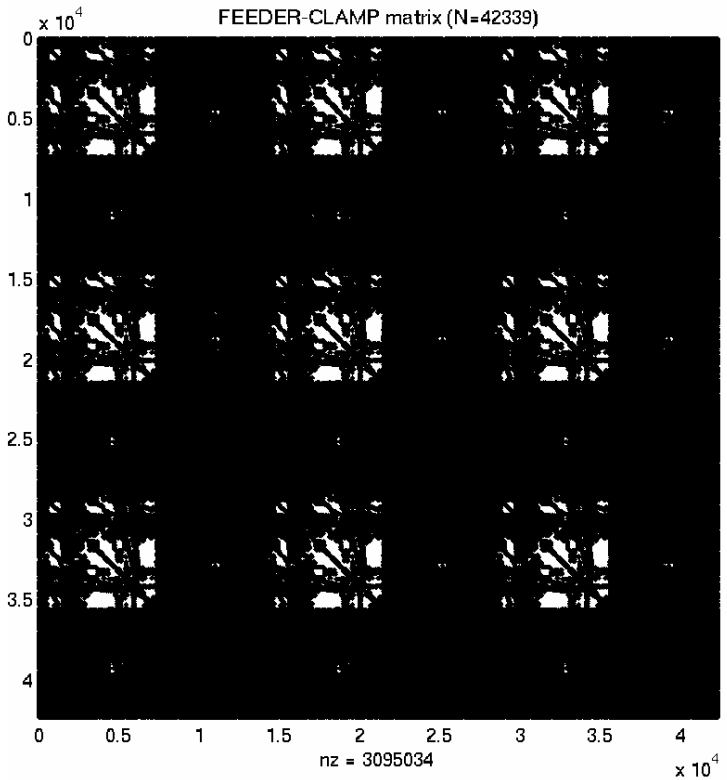
$$Deg(y) = \min_{x \in \{\mathcal{N}-S\}} Deg(x)$$

   Number the node y next and set $T \leftarrow S \cup \{y\}$
3. (*Degree update*) $Deg(u) \leftarrow |Reach(u, T)| \;\; \forall u \in \mathcal{N} - T$
4. (*loop or stop*) If $T = \mathcal{N}$ stop. Otherwise set $S \leftarrow T$ and go to 2

Minimum degree algorithm

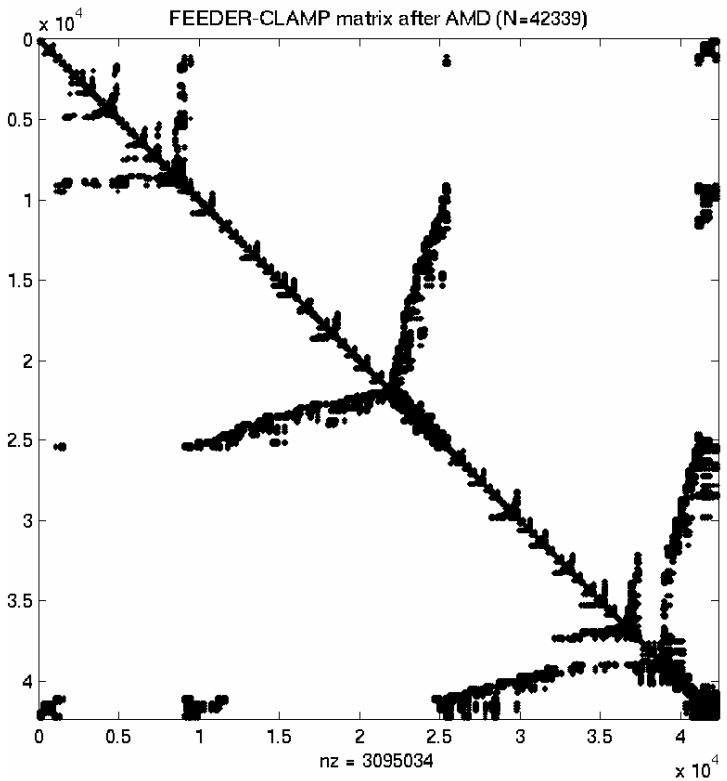
The implementation of the Minimum degree algorithm takes advantage of several other results that improve the speed and reduce the complexity.
It is possible to partially update the degrees using special heuristics
A good example is the APPROXIMATE MINIMUM DEGREE used in Matlab and in HSL package MA57. The use of the AMD can be extended to matrix which are symmetric and indefinite (MA57)
(See Davis, Amestoy, and Duff SIMAX, 1996 , and Duff RAL-TR-2002-024, 2002)

Minimum degree algorithm



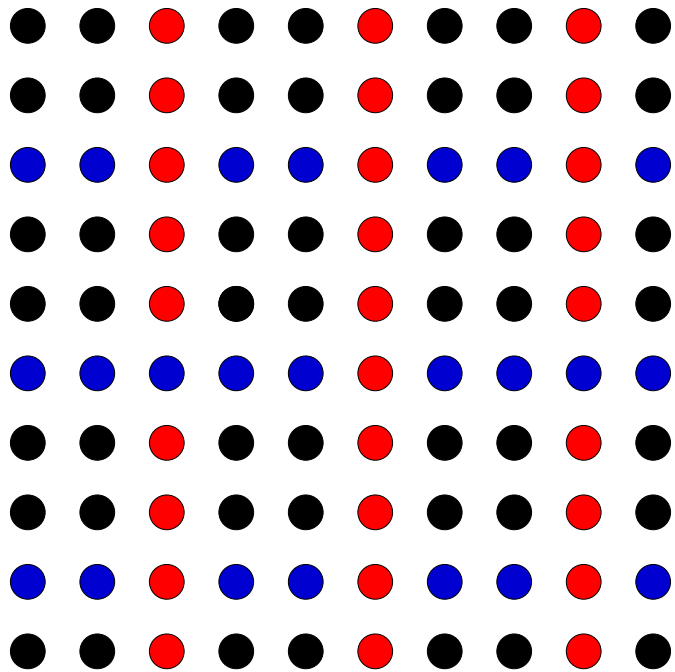
A with $n = 42339$ $nz(\mathbf{A}) = 3095034$ and $\delta = 0.0017$

Minimun degree algorithm



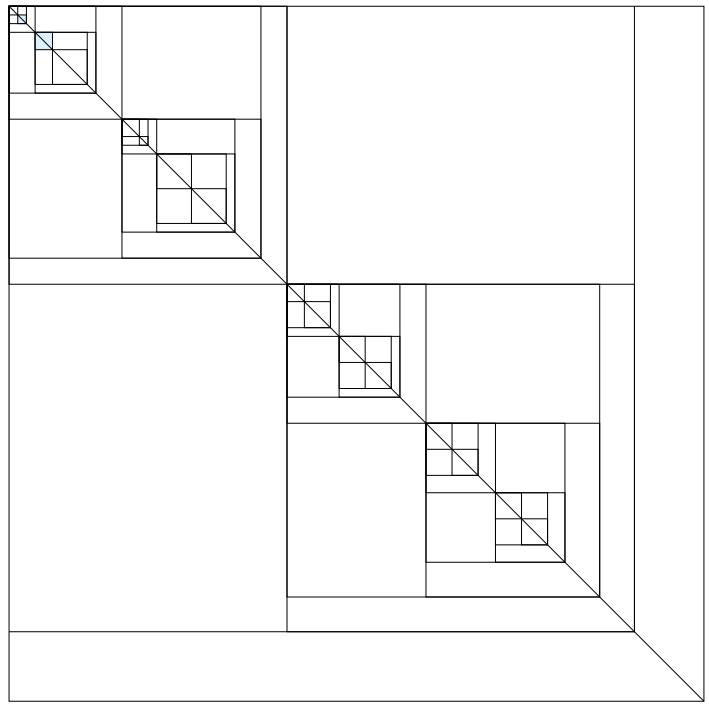
after AMD

Nested Dissection



Nested Dissection Partition of the Graph of a Finite Difference problem matrix on a 10×10 mesh

Nested Dissection



Nested Dissection Permutation of a Finite Difference problem matrix on a 10×10 mesh

Nested Dissection: storage requirement

The number of nonzeros in the factor $\mathbf{L}$ of a matrix $\mathbf{A} \in \mathrm{R}^N$ associated to a $n \times n$ grid ($N = n^2$) is

$$31n^2 \log_2(n)/4 + \mathcal{O}(n^2)$$

Nested Dissection: complexity

The number of operations required to factor $\mathbf{L}$ of a matrix $\mathbf{A} \in \mathrm{R}^N$ associated to a $n \times n$ grid ($N = n^2$) is

$$829n^3/84 + \mathcal{O}(n^2 \log_2(n))$$

Fiedler vectors

How can we generalise the process of Nested Dissection to a general symmetric matrix?
$\mathcal{P}_A - \mathbf{I}$ is the adjacency matrix of a the graph supporting the matrix $\mathbf{A}$. Let us denote by $\mathbf{E_A}$ the incidence matrix of the graph. We can interpret $\mathbf{E}$ as a discrete divergence operator.
We denote by

$$\mathbf{L} = \mathbf{E_A}\mathbf{E_A}^T$$

the Graph-Laplacian .

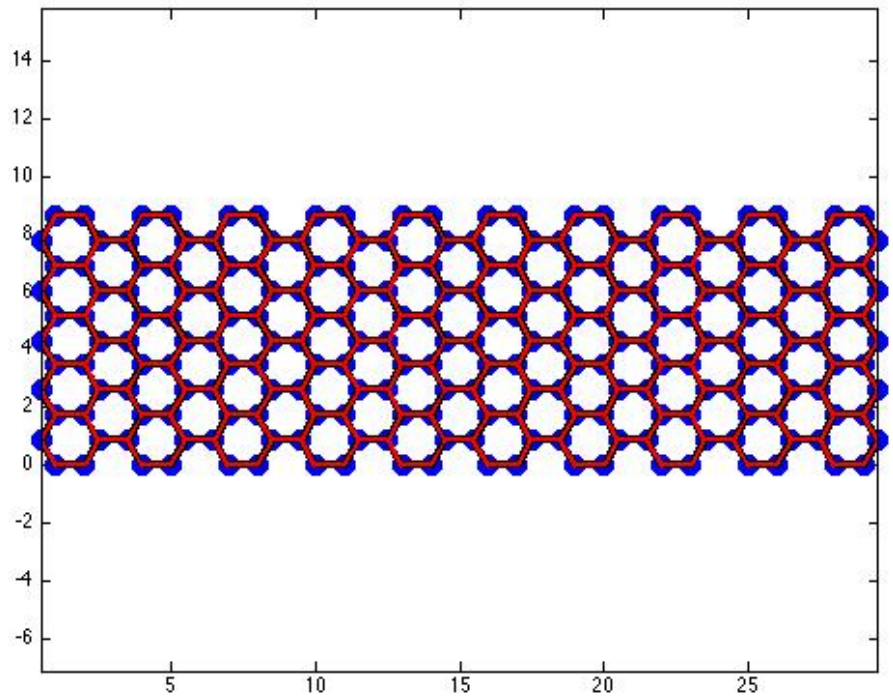
Fiedler vectors

L is a Laplacian with Neumann conditions!! Therefore its smallest eigenvalue is zero. Le assume that the eigenvalues $\lambda_i(\mathbf{L})$ are ordered in increasing order.

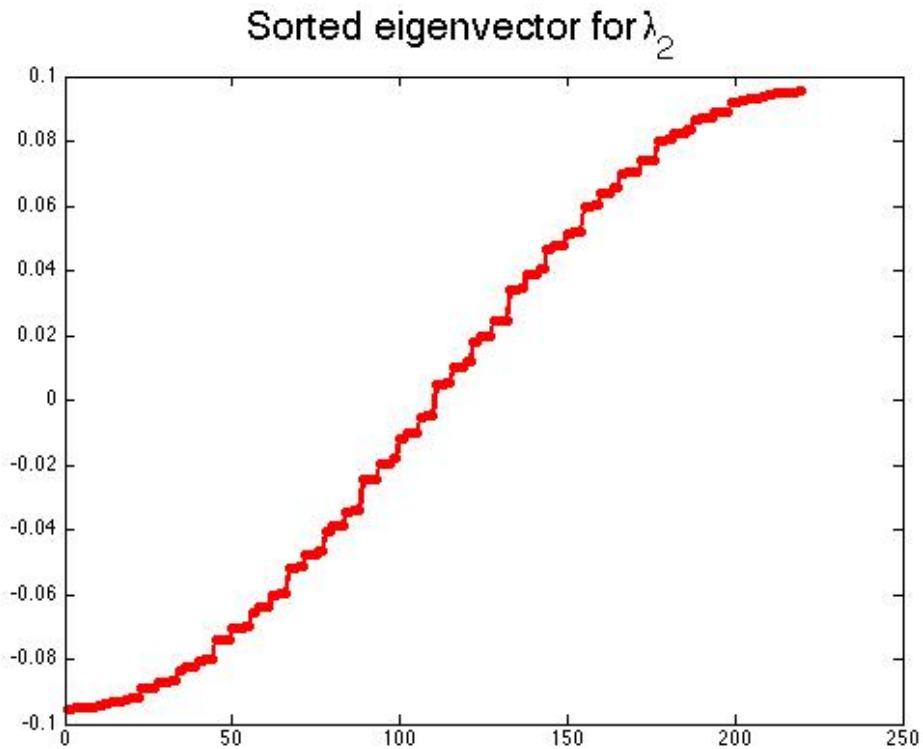
Fiedler vectors

L is a Laplacian with Neumann conditions!! Therefore its smallest eigenvalue is zero. Le assume that the eigenvalues $\lambda_i(\mathbf{L})$ are ordered in increasing order.
What about $\lambda_2(\mathbf{L})$?

Fiedler vectors



Fiedler vectors



Sorted eigenvector for $\lambda_2(\mathbf{L})$

Fiedler vectors

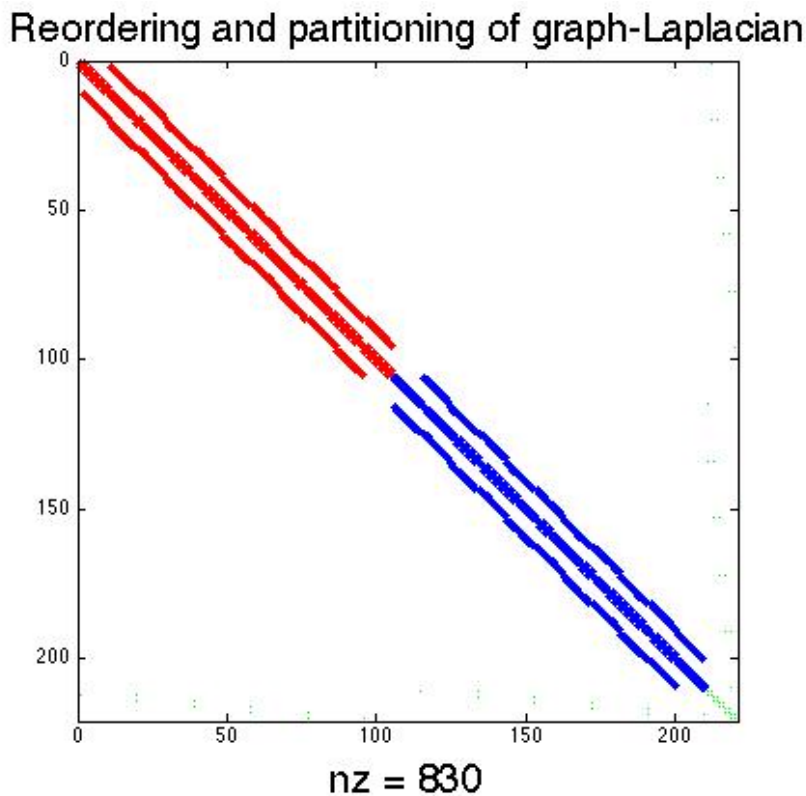
We have positive and negative values in each node. Some of them are small in absolute values! .

Fiedler vectors

Let us separate the nodes with values $\leq -\tau$ from the nodes with values $\geq \tau$, and from the nodes having values $\in (-\tau, \tau)$.

Fiedler vectors

. We can then reorder the nodes and we have



Permuted graph-Laplacian

Fiedler vectors

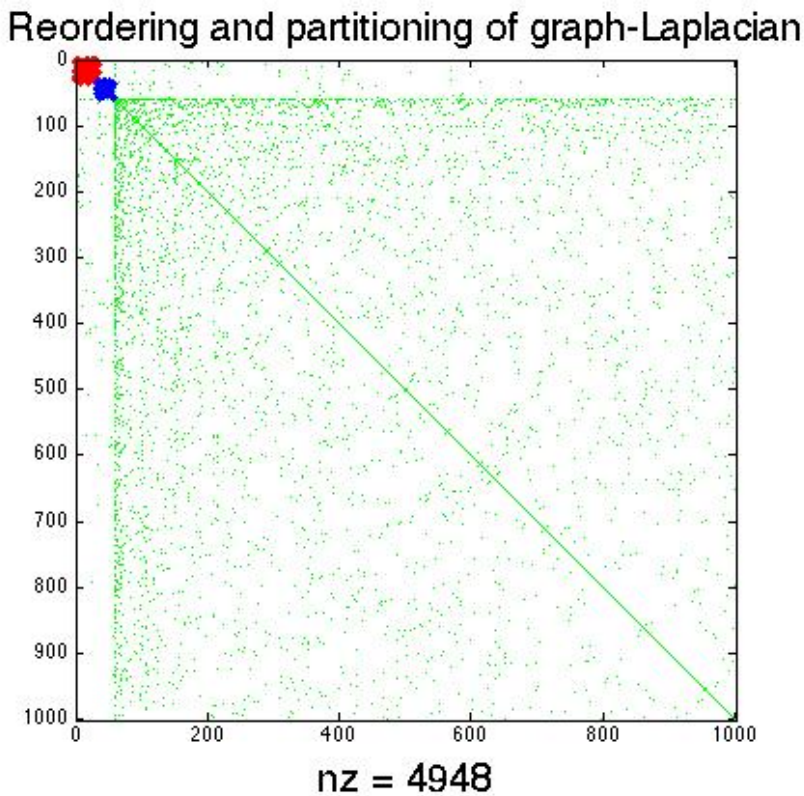
In the previous example we had a separator that is $\sqrt{n}$ with n the order of $\mathbf{L}$.
More generally, if such a separator does exist not only for $\mathbf{L}$ but also for the the red block and the blue block then we can iterate the process on each of them and have a
GENERALISED NESTED DISSECTION (see METIS).

Fiedler vectors

UNFORTUNATELLY it is not always the case!

Fiedler vectors

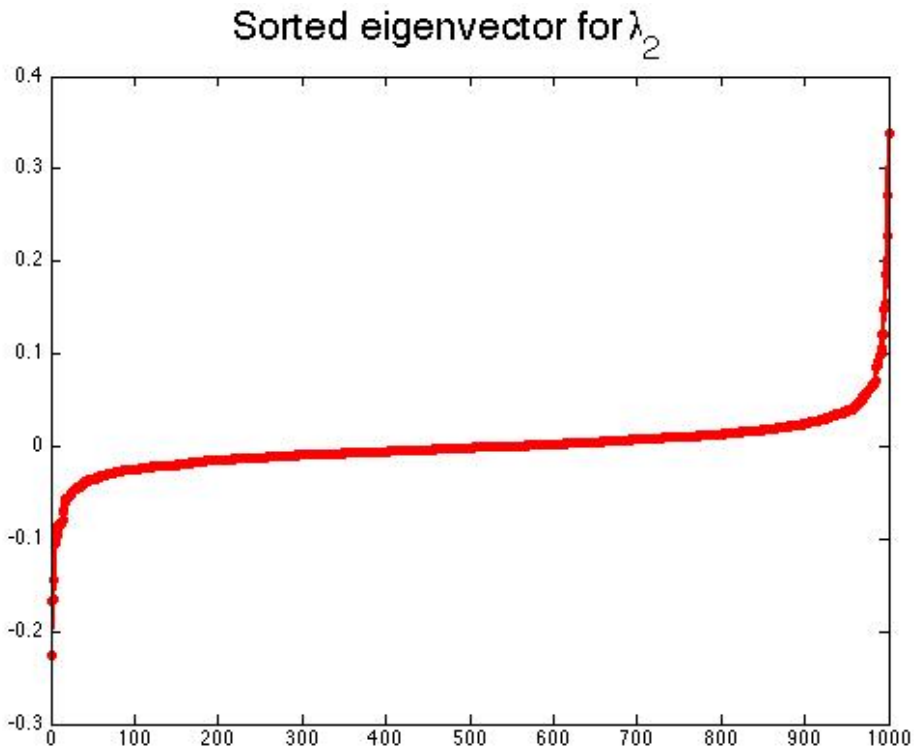
UNFORTUNATELLY it is not always the case!



Permuted graph-Laplacian

Fiedler vectors

UNFORTUNATELLY it is not always the case!



Sorted eigenvector for $\lambda_2(\mathbf{L})$

The unsymmetric matrices: Markowitz criterion

Analogously to the minimum degree criterion, we look at the Schur complement.

$$
\begin{aligned}
\mathbf{A} &= \begin{bmatrix} d_1 & \mathbf{w}_1^T \\ \mathbf{v}_1 & \mathbf{H}_1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 \\ \mathbf{v}_1/d_1 & I_{n-1} \end{bmatrix} \begin{bmatrix} d_1 & \mathbf{w}_1^T \\ 0 & \mathbf{S_1} \end{bmatrix} \\
\mathbf{S_1} &= \mathbf{H}_1 - \frac{\mathbf{v}_1 \mathbf{w}_1^T}{d_1}
\end{aligned}
$$

Repeat the operation on $\mathbf{S_1}$. $\mathbf{L} = \mathbf{L}_{n-1} \ldots \mathbf{L}_1$ and $\mathbf{A} = \mathbf{LU}$

The unsymmetric matrices: Markowitz criterion

Let $r_i^{(k)}$ be the number of entries in row i of $\mathbf{S_k}$ and $c_j^{(k)}$ be the number of entries in column j of $\mathbf{S_k}$. The Markowitz criterion chooses as pivot the entry a_{ij}^k in $\mathbf{S_k}$ such that $(r_i^{(k)} - 1)(c_j^{(k)} - 1)$ is minimized.
In the symmetric case $r_i^{(k)} = c_i^{(k)}$ and we obtain the minimum degree rule.
Moreover, we need to control the numerical growth on the entries during factorization. Thus, we restrict the Markowitz selection to those pivot candidates that satisfy the inequality

$$|a_{kk}^{(k)}| \geq u|a_{ik}^{(k)}| \qquad i \geq k \qquad 0 < u \leq 1$$

Analysis phase and Factorization Phase

The reordering algorithms presented are heuristics:
to find the permutation that minimizes the fill-in for a symmetric matrix is a NP-complete problem
First, we ANALYSE the data structure:

- Reordering
- Estimate if the LU factorization feasible (memory requirement)
- Elimination tree

Then, we can FACTORIZE

Frontal Methods

$$\mathbf{A} = \sum_{l=l}^{m} \mathbf{A}^{[l]}$$

where $\mathbf{A}^{[l]}$ has nonzeros only in few rows and columns (corresponding to an element in the mesh if we are in a finite-element framework)
Assembling

$$a_{ij} \leftarrow a_{ij} + a_{ij}^{[l]}$$

Factorization

$$a_{ij} \leftarrow a_{ij} - a_{ip}(a_{pp})^{-1}a_{pj}$$

can be performed as soon all the terms are assembled

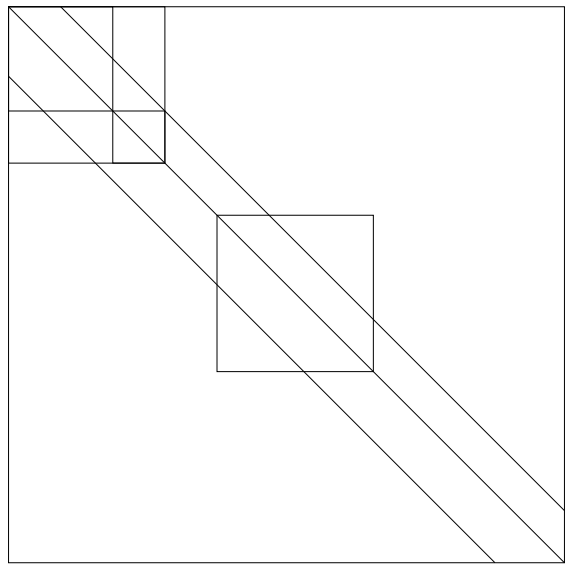
Frontal Methods

We can work only on a small matrix **F** , the Frontal Matrix, that is dense but with smaller dimensions compared to **A**

$$\mathbf{F} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}$$

B is square of order k and **E** is square of order r: $\mathbf{F} \in \mathrm{R}^{(k+r)\times(k+r)}$
$k << r$, typically $k = 10, 20$ and $r = 200, ..., 500$

Frontal Methods



Front as a window

Multifrontal Methods

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

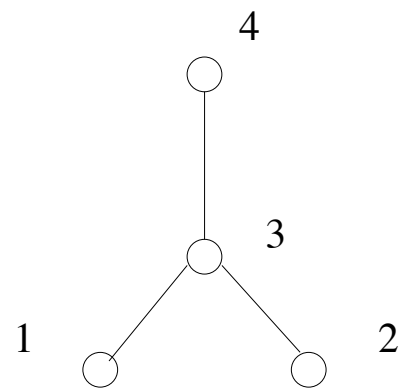
Assemble the first row and column

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

Perform the operation on this front and then assemble the second row and column and factorize.
First front and second front are independent of each other:
we can do the operation in parallel

Multifrontal Method: elimination tree

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



Elimination tree

Numerical pivot

- Numerical Pivot in sparse solvers
- Scaling
- A-Posteriori sparse backward error analysis
- Condition number estimators
- Iterative refinement

Gaussian elimination

Theorem Let assume that $\mathbf{A}$is an $n \times n$ matrix of floating point numbers. If no zero pivots are found during the Gauss process, then the computed matrices $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ satisfy

$$\begin{aligned} \hat{\mathbf{L}}\hat{\mathbf{U}} &= \mathbf{A} + \mathbf{E} \\ |\mathbf{E}| &\leq 3(n-1)\epsilon(|\mathbf{A}| + |\hat{\mathbf{L}}||\hat{\mathbf{U}}|) + \mathcal{O}(\epsilon^2) \end{aligned}$$

We would like that

$$|\mathbf{A}| \approx |\hat{\mathbf{L}}||\hat{\mathbf{U}}|$$

Numerical Pivot in sparse solvers

We know that the partial pivot is not optimal and we have examples where the $|\mathbf{L}||\mathbf{U}|$ is much larger than $|\mathbf{A}|$

$$\mathbf{A} = \begin{bmatrix} 1 & & & 1 \\ -1 & 1 & & 1 \\ \vdots & \vdots & \vdots & \vdots \\ -1 & \dots & -1 & 1 \end{bmatrix}$$

in the $\mathbf{U}$ factor the last column has entries that grow as $2^i,\ i = 1, \dots, n-1$

Frontal Methods

In the Frontal Matrix, that is dense but with smaller dimensions compared to **A**, we can choose the pivot only in fully assembled **B**.

$$\mathbf{F} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{D} & \mathbf{E} \end{bmatrix}$$

If we detect a larger entry in **D** (or in **C**), we have only two options:

- to delay the pivot moving the corresponding row and column to **E** with the hope that further assembling operations will cure the problem. In the worst case, we will be able to take care of this pivot in the last matrix
- to correct the value of the pivot adding to it a constant value β:
  - $\beta = \sqrt{\epsilon}||\mathbf{A}||$
  - $\beta = ||\mathbf{A}||$

Frontal Methods

In both cases, we need to recover the solution of $\mathbf{Au} = \mathbf{b}$.
The previous corrections can be seen as perturbations of the matrix $\mathbf{A}$ reordered to reduce fill-in. The computed factor are the computed factors of

$$\mathbf{A} + \sum_{i=1}^{k} \beta_i \mathbf{e}_i \mathbf{e}_i^T$$

The first technique allows more pivot corrections than the second one. If $\mathbf{A}$ has a condition number smaller than $1/\sqrt{\epsilon}$, we can use an iterative method to recover the solution
In the second case, we can use the Sherman-Morrison formula to correct the computed solution of the perturbed problem and recover $\mathbf{u}$

Scaling

It is good practice to scale the matrix $\mathbf{A}$ to obtain a new matrix $\mathbf{B}$ where the nonzero entries are between -1 and 1.
Given $\mathbf{D}_1$ and $\mathbf{D}_2$ diagonal matrices we have

$$\mathbf{B} = \mathbf{D}_1 \mathbf{A} \mathbf{D}_2$$

and we solve

$$\mathbf{B}\mathbf{y} = \mathbf{c} \qquad \mathbf{y} = \mathbf{D}_2^{-1}\mathbf{x} \qquad \text{and } \mathbf{c} = \mathbf{D}_1\mathbf{b}$$

Scaling

There are several approach

- equilibration of the data (Curtis and Reid)
- $(\mathbf{D}_1)_{ii} = 1/||\mathbf{A}_{i\bullet}||_1$ or/and $(\mathbf{D}_2)_{ii} = 1/||\mathbf{A}_{\bullet i}||_1$
- $(\mathbf{D}_1)_{ii} = 1/||\mathbf{A}_{i\bullet}||_\infty$ or/and $(\mathbf{D}_2)_{ii} = 1/||\mathbf{A}_{\bullet i}||_\infty$
- Compute $\mathbf{D}_1$ and $\mathbf{D}_2$ such that $|\mathbf{B}|$ is doubly stochastic (Ruiz 2002)

All these technique improve the quality of the LU factorization and decrease the need of choosing numerical pivots different from the ones *optimal* for controlling the fill-in

Rigal-Gaches (1967) theorem

$$\left.\begin{array}{l}\exists \Delta\mathbf{A}, \exists \delta\mathbf{b} \text{ such that:} \\ (\mathbf{A}+\Delta\mathbf{A})\tilde{\mathbf{u}} = (\mathbf{b}+\delta\mathbf{b}) \\ \text{and } \|\Delta\mathbf{A}\|_{\vec{k},\vec{p}} \leq \mathbf{S} \in \mathbf{R}^{k\times p}, \|\delta\mathbf{b}\|_{\vec{k}} \leq \mathbf{t} \in \mathbf{R}^{k}\end{array}\right\} \Leftrightarrow \begin{cases}\|\mathbf{r}\|_{\vec{k}} \leq \mathbf{S}\|\tilde{\mathbf{u}}\|_{\vec{p}} + \mathbf{t} \\ \text{where } \mathbf{r} \text{ is defined by} \\ \mathbf{r} = \mathbf{A}\tilde{\mathbf{u}} - \mathbf{b}\end{cases}$$

Rigal-Gaches (1967) theorem: component-wise version

If we use $|\bullet|$ as Hypernorm, we have a *component-wise* version

$$\left.\begin{array}{l}\exists \Delta\mathbf{A}, \exists \delta\mathbf{b} \text{ such that:} \\ (\mathbf{A}+\Delta\mathbf{A})\tilde{\mathbf{u}} = (\mathbf{b}+\delta\mathbf{b}) \qquad \text{and} \\ |\Delta\mathbf{A}| \leq \omega|\mathbf{A}| \in \mathbb{R}^{n\times n}, |\delta\mathbf{b}| \leq \omega|\mathbf{b}| \in \mathbb{R}^{n}\end{array}\right\} \Leftrightarrow \begin{cases} |\mathbf{r}| \leq \omega\{|\mathbf{A}||\tilde{\mathbf{u}}| + |\mathbf{b}|\} \\ \text{where } \mathbf{r} \text{ is defined by} \\ \mathbf{r} = \mathbf{A}\tilde{\mathbf{u}} - \mathbf{b} \end{cases}$$

The smallest ω satisfying the theorem (assuming $0/0 = 0$) is

$$\omega = \min_i \frac{(|\mathbf{A}\tilde{\mathbf{u}} - \mathbf{b}|)_i}{(|\mathbf{A}||\tilde{\mathbf{u}}| + |\mathbf{b}|)_i}$$

$$\frac{\|\tilde{\mathbf{u}} - \mathbf{u}\|_\infty}{\|\mathbf{u}\|_\infty} \leq \omega \frac{\||\mathbf{A}^{-1}|(|\mathbf{A}||\mathbf{u}| + |\mathbf{b}|)\|_\infty}{\|\mathbf{u}\|_\infty}$$

Condition number estimators

The

$$cond(\mathbf{A},\mathbf{u}) = \frac{\||\mathbf{A}^{-1}|(|\mathbf{A}||\mathbf{u}| + |\mathbf{b}|)\|_\infty}{\|\mathbf{u}\|_\infty}$$

is the Condition Number (also known as the Bauer-Skeel condition number) of the Problem

$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

Obviously, the computation of $|\mathbf{A}^{-1}|$ is totally unfeasible for large sparse matrices. The pattern of the inverse will be structurally full. We would like to approximate the value in $\mathcal{O}(n^2)$ operation.
The expression $\||\mathbf{A}^{-1}|\mathbf{d}\|_\infty$ with $d \geq 0$ can be rewritten using $\mathbf{D} = diag(\mathbf{d})$ and $\mathbf{e} = [1, 1, \ldots, 1]^T$ as follows

$$\||\mathbf{A}^{-1}|\mathbf{d}\|_\infty = \||\mathbf{A}^{-1}|\mathbf{D}\mathbf{e}\|_\infty = \||\mathbf{A}^{-1}\mathbf{D}|\mathbf{e}\|_\infty = \|\mathbf{A}^{-1}\mathbf{D}\|_\infty$$

Then, we need a *good* estimator for the $\|\mathbf{B}\|_\infty$ or, equivalently, for the $\|\mathbf{B}\|_1$ ($\mathbf{B}$ given $n \times n$ matrix).

$$\|\mathbf{B}\|_\infty = \|\mathbf{B}^T\|_1$$

Condition number estimators: Hager's algorithm

In LAPACK and in MC75 (HSL 2002) we have good implementation of Hager's algorithm (SISSC 1984) for the estimate of the 1-norm of a matrix $\mathbf{A}$ when we are only able to compute the matrix-vector product $\mathbf{Av}$.

Condition number estimators: Hager's algorithm

Algorithm Given $\mathbf{A} \in \mathrm{R}^{n\times n}$ the algorithm computes γ and $\mathbf{v} = \mathbf{A}\mathbf{w}$ s.t. $\gamma \leq \|\mathbf{A}\|_1$ with $\|\mathbf{v}\|_1/\|\mathbf{w}\|_1 = \gamma$

$\mathbf{v} = \mathbf{A}(n^{-1}\mathbf{e})$

if $n = 1$, quit with $\gamma = |v_1|$, end

$\gamma = ||\mathbf{v}||_1$, $\xi = sign(\mathbf{v})$, $\mathbf{x} = \mathbf{A}^T\xi$, $k = 2$

repeat

$j = \min\{i : |xv_i| = ||\mathbf{x}||_\infty\}$

$\mathbf{v} = \mathbf{A}\mathbf{e}_j$, $\bar{\gamma} = \gamma$, $\gamma = ||\mathbf{v}||_1$

if $sign(\mathbf{v}) = \xi$ or $\gamma \leq \bar{\gamma}$, goto (*), end

$\xi = sign(\mathbf{v})$, $\mathbf{x} = \mathbf{A}^T\xi$, $k = k + 1$

until $(||\mathbf{x}||_\infty = x_j$ or $k > 5$

(*) $x_i = (-1)^{i+1}\left(1 + \frac{i-1}{n-1}\right)$ $i = 1, \ldots, n$

$\mathbf{x} = \mathbf{A}x$

if $2||\mathbf{x}||_1/(3n) > \gamma$ then

$\mathbf{v} = \mathbf{x}$, $\gamma = 2||\mathbf{x}||_1/(3n)$

end if

Condition number estimators

The algorithm is very reliable in practice. It is very rare for the estimate to be more than three time smaller than the the actual norm.

Condition number estimators

The algorithm is very reliable in practice. It is very rare for the estimate to be more than three time smaller than the the actual norm.
Nevertheless, it can give the wrong answer. For the following class of matrices the algorithm returns the WRONG value 1

$$\mathbf{A}(\theta) = \mathbf{I} + \theta\mathbf{P}$$

where $\mathbf{P} = \mathbf{P}^T$, $\mathbf{Pe} = 0$, $\mathbf{Pe}_1 = 0$, and $\mathbf{Px} = 0$

Condition number estimators: LINPACK

An alternative approach is used in LINPACK. The original idea is in Cline, Moler, Stewart, and Wilkinson (1978). The algorithm applies to triangula matrices and can be used in sevarl other situations (Bischof (1990) used one variants for rank revealing in QR, Duff and Voemel (2000) used other variants to estimate the 2-norm condition number)

Condition number estimators: LINPACK

An alternative approach is used in LINPACK. The original idea is in Cline, Moler, Stewart, and Wilkinson (1978). The algorithm applies to triangula matrices and can be used in sevarl other situations (Bischof (1990) used one variants for rank revealing in QR, Duff and Voemel (2000) used other variants to estimate the 2-norm condition number) Given the triangular matrix $^{\mathsf{T}} \in \mathbf{R}^{n\times n}$

1. Choose a vector $\mathbf{d}$ s.t. $||\mathbf{y}||$ is large as possible relative to $||\mathbf{d}||$, where $\mathbf{T}^T\mathbf{y} = \mathbf{d}$
2. Solve $\mathbf{T}\mathbf{x} = \mathbf{y}$
3. Estimate $||\mathbf{T}^{-1}|| \approx ||\mathbf{x}||/||\mathbf{y}||$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$
fixed precision
Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$
- Solve $\mathbf{Ad} = \mathbf{r}$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$
fixed precision
Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$ $\quad |fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
- Solve $\mathbf{Ad} = \mathbf{r}$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$
fixed precision
Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$ $\quad |fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
- Solve $\mathbf{Ad} = \mathbf{r}$ $\quad (\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b} \;\; \|\mathbf{A}^{-1}\mathbf{E}\|_\infty < 1$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$
fixed precision
Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$ $\quad |fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
- Solve $\mathbf{Ad} = \mathbf{r}$ $\quad (\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b} \;\; \|\mathbf{A}^{-1}\mathbf{E}\|_\infty < 1$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$ $\quad |fl(\hat{\mathbf{u}}) - (\hat{\mathbf{u}} + \mathbf{d})| \leq \epsilon(|\hat{\mathbf{u}} + \mathbf{b}|)$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$

fixed precision mixed precision

Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$ $\quad |fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
- Solve $\mathbf{Ad} = \mathbf{r}$ $\quad (\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b} \;\; \|\mathbf{A}^{-1}\mathbf{E}\|_\infty < 1$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$ $\quad |fl(\hat{\mathbf{u}}) - (\hat{\mathbf{u}} + \mathbf{d})| \leq \epsilon(|\hat{\mathbf{u}} + \mathbf{b}|)$

Iterative refinement

Given $\hat{\mathbf{u}}$ and the system $\mathbf{Au} = \mathbf{b}$

fixed precision mixed precision

Repeat until convergence

- Compute $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{u}}$ $\quad |fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
  $|fl(\mathbf{r}) - \mathbf{r}| \leq \epsilon(|\mathbf{A}\hat{\mathbf{u}} - \mathbf{b}|) + \epsilon^2(|\mathbf{A}||\hat{\mathbf{u}}| + |\mathbf{b}|)$
- Solve $\mathbf{Ad} = \mathbf{r}$ $\quad (\mathbf{A} + \mathbf{E})\mathbf{y} = \mathbf{b} \;\; \|\mathbf{A}^{-1}\mathbf{E}\|_\infty < 1$
- Update $\hat{\mathbf{u}} = \hat{\mathbf{u}} + \mathbf{d}$ $\quad |fl(\hat{\mathbf{u}}) - (\hat{\mathbf{u}} + \mathbf{d})| \leq \epsilon(|\hat{\mathbf{u}} + \mathbf{b}|)$

Iterative refinement

Theorem 1 Let iterative refinement be applied to the nonsingular system $\mathbf{Au} = \mathbf{b}$ of order n, using LU factorization. Let $\eta = \epsilon \| |\mathbf{A}^{-1}| |\hat{\mathbf{L}}| |\hat{\mathbf{U}}| \|_\infty$, where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the computed LU factors of $\mathbf{A}$. Then, provided η is sufficiently less than 1, iterative refinement reduces the error by a factor η at each step until

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lessapprox 2n\epsilon \frac{\| |\mathbf{A}^{-1}| (|\mathbf{A}||\mathbf{u}| + |\mathbf{b}|) \|_\infty}{\|\mathbf{u}\|_\infty}$$

Iterative refinement

Theorem 1 Let iterative refinement be applied to the nonsingular system $\mathbf{Au} = \mathbf{b}$ of order n, using LU factorization. Let $\eta = \epsilon \| |\mathbf{A}^{-1}||\hat{\mathbf{L}}||\hat{\mathbf{U}}| \|_\infty$, where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the computed LU factors of $\mathbf{A}$. Then, provided η is sufficiently less than 1, iterative refinement reduces the error by a factor η at each step until

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \lessapprox 2n\epsilon \frac{\| |\mathbf{A}^{-1}|(|\mathbf{A}||\mathbf{u}| + |\mathbf{b}|)\|_\infty}{\|\mathbf{u}\|_\infty}$$

Theorem 2 Let iterative refinement be applied to the nonsingular system $\mathbf{Au} = \mathbf{b}$ of order n, using LU factorization and with residual computed in double the working precision. Let $\eta = \epsilon \| |\mathbf{A}^{-1}||\hat{\mathbf{L}}||\hat{\mathbf{U}}| \|_\infty$, where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the computed LU factors of $\mathbf{A}$. Then, provided η is sufficiently less than 1, iterative refinement reduces the error by a factor η at each step until

$$\frac{\|\mathbf{u} - \hat{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty} \approx \epsilon$$

Iterative refinement

Theorem 3 Let iterative refinement be applied to the nonsingular system $\mathbf{Au} = \mathbf{b}$ of order n, using LU factorization and with residual computed in the working precision. Let $\eta = \epsilon \| |\mathbf{A}^{-1}| |\hat{\mathbf{L}}| |\hat{\mathbf{U}}| \|_\infty$, where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the computed LU factors of $\mathbf{A}$. Then, provided η is sufficiently less than 1, iterative refinement reduces the residual by a factor η at each step until

$$\omega = \min_i \frac{(|\mathbf{A}\tilde{\mathbf{u}} - \mathbf{b}|)_i}{(|\mathbf{A}||\tilde{\mathbf{u}}| + |\mathbf{b}|)_i} \approx \epsilon$$

Iterative refinement

Theorem 3 Let iterative refinement be applied to the nonsingular system $\mathbf{Au} = \mathbf{b}$ of order n, using LU factorization and with residual computed in the working precision. Let $\eta = \epsilon \| |\mathbf{A}^{-1}||\hat{\mathbf{L}}||\hat{\mathbf{U}}| \|_\infty$, where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the computed LU factors of $\mathbf{A}$. Then, provided η is sufficiently less than 1, iterative refinement reduces the residual by a factor η at each step until

$$\omega = \min_i \frac{(|\mathbf{A}\tilde{\mathbf{u}} - \mathbf{b}|)_i}{(|\mathbf{A}||\tilde{\mathbf{u}}| + |\mathbf{b}|)_i} \approx \epsilon$$

The definition of ω and Theorem 3 must be slightly modified if $\mathbf{b}$ is also sparse or has very small entries (see Arioli, Demmel, and Duff (1989) SIMAX)
A more general result can be proved (Jankowski and Wozniakowski 1977). The norm-wise backward convergence of the IRA is proved for an arbitrary linear solver in fixed precision as long as the solver is not too unstable and $\mathbf{A}$ is not too ill-conditioned.

Plan B

If IRA fails to converge we have still FGMRES

Existing software

- MA41 (HSL) or MUPS (Amestoy and L'Excellent Toulouse):symmetric pattern multifrontal parallel
- MA48 (HSL) unsymmetric matrices (includes IRA and error estimators) (parallel version)
- MA49 (HSL) sparse QR for least-squares problems.
- MA57 (HSL) symmetric indefinite: LDL^T factorization
- MA52 (HSL) out-of-core multiple front
- MA72 (HSL) out-of-core frontal method for finite-element matrices
- MC25 (HSL) BTF
- MC64 (HSL) Scaling and maximum transversal maximizing the smallest diagonal entry in abs. value
- MC75 (HSL) IRA
- SuperLU (Li, Demmel) unsymmetric case with static pivot
- more .... look in the book *Direct Methods for Sparse Matrices* Duff, Erisman, and Reid (second Ed. soon) and

HSL http://www.hsl.rl.ac.uk

Final remarks

- Scale the data
- Choose a good reordering reducing the fill-in
- Factorize avoiding as much as possible the numerical pivot: decrease the threshold or use carefully a static pivot
- Control the quality of the factorization $\eta = \epsilon \| |\mathbf{A}^{-1}| |\hat{\mathbf{L}}| |\hat{\mathbf{U}}| \|_\infty$ must be less than 1
- Use IRA (or FGMRES)
- Estimate the final error ω $cond(\mathbf{A}, \mathbf{u})$