## *Multifrontal methods*

- Start with the frontal method.
- Recall: Finite element matrix:

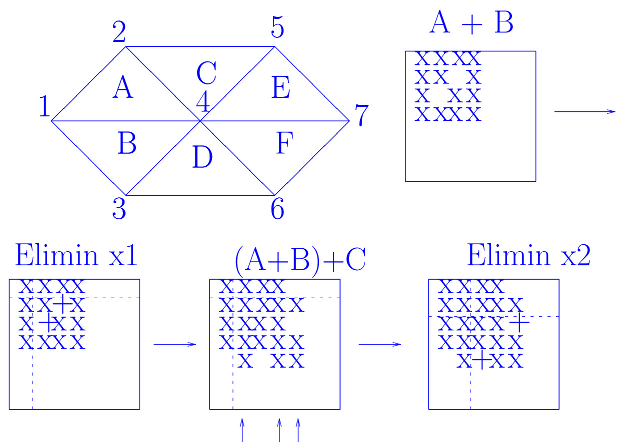
$$\boldsymbol{A} = \sum \boldsymbol{A}^{[e]}$$

$\boldsymbol{A}^{[e]}$ = element matrix associated with element $\boldsymbol{e}$.

- An old idea: Execute Gaussian elimination as the elements are being assembled
- This is called *the frontal method*
- Very popular among finite element users: saves storage



## *The origin: Frontal method*




- Elimination of $x_1$ creates an update matrix

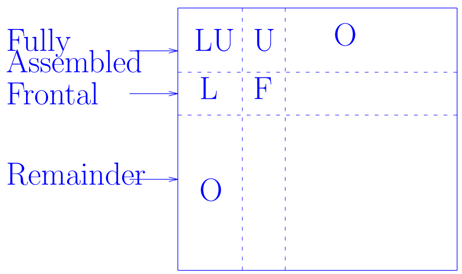


- Matrix has 3 parts:

1) Fully assembled (no longer modified)

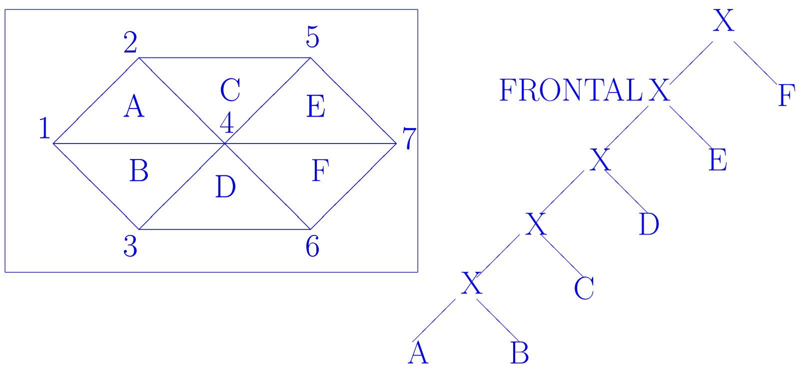
2) Frontal matrix: undergoes assembly + updates

3) Remainder: not accessed yet.



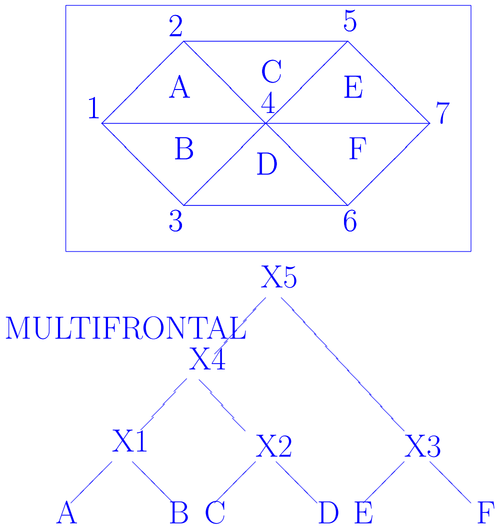



**Assembly tree:** - analogue to elimination tree




- Can proceed from several incoupled elements at the same time $\longrightarrow$ multifrontal technique [Duff & Reid, 1983]

## Assembly tree for Multifrontal Method

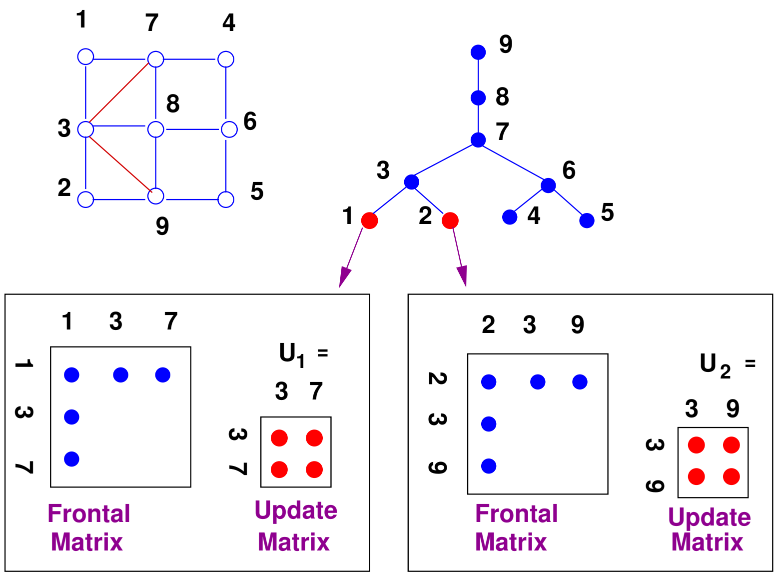


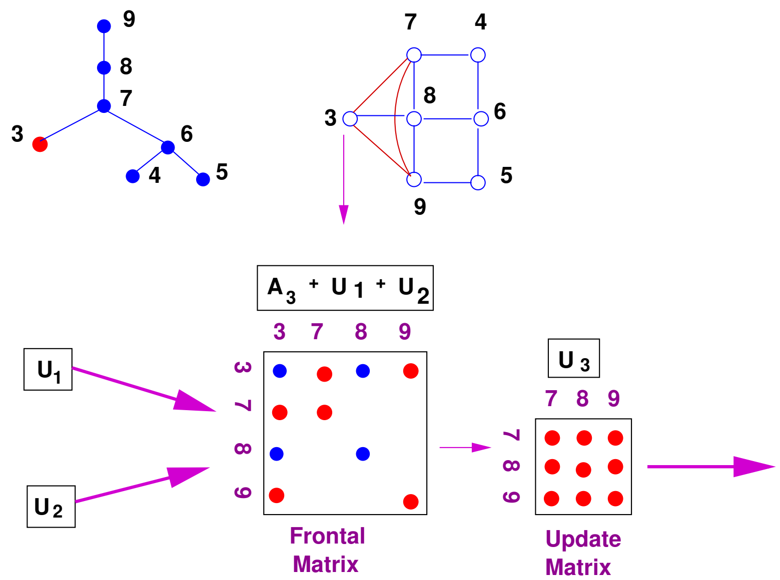



## *Multifrontal methods: extension to general matrices*

- Elimination tree replaces assembly tree
- Proceed in post-order traversal of elimination tree in order not to violate task dependencies.
- When a node is eliminated an update matrix is created.
- This matrix is passed to the parent which adds it to its frontal matrix.
- Requires a stack of pending update matrices
- Update matrices popped out as they are needed
- Typically implemented with nested dissection ordering
- More complex than a left-looking algorithm





## Eliminating nodes 1 and 2: What happens on matrix

$$
\begin{array}{l}
\left[\begin{array}{c|c|c|ccc|c|c|c}
1 & & \star & & & & \star & & \\
\hline
 & 2 & \star & & & & & & \star \\
\hline
\star & \star & 3 & & & & \blacksquare & \star & \blacksquare \\
\hline
 & & & 4 & & \star & \star & & \\
 & & & & 5 & \star & & & \star \\
 & & & \star & \star & 6 & & \star & \\
\hline
\star & & \blacksquare & \star & & & 7 & \star & \\
\hline
 & & \star & & & \star & \star & 8 & \star \\
\hline
 & \star & \blacksquare & & \star & & & \star & 9
\end{array}\right]
\end{array}
\begin{array}{l}
\\ \\ \leftarrow U_1(3,:) \leftarrow U_2(3,:) \\ \\ \\ \\ \leftarrow U_1(7,:) \\ \\ \qquad\qquad \leftarrow U_2(9,:)
\end{array}
$$



## *Supernodes*

➤ In GE, contiguous columns tend to inherit the same pattern as the columns from they are updated $\rightarrow$ Many columns will have same sparsity pattern.

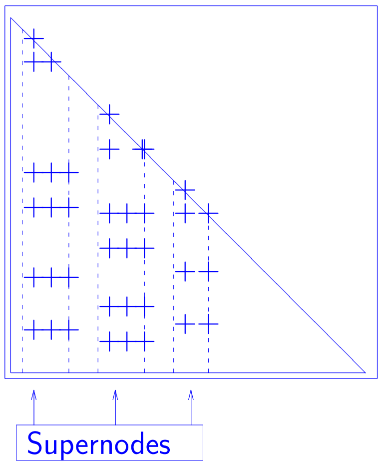
A supernode = a set of contiguous columns in the Cholesky factor $L$ which have the same sparsity pattern.

The set $\{j, j+1, ..., j+s\}$ is a supernode if

$$NZ(L_{*,k}) = NZ(L_{*,k+1}) \bigcup \{k+1\} \quad j \le k < j+s$$

where $NZ(L_{*,k})$ is nonzero set of column $k$ of $L$.

## Supernodes






Other terms used: Mass elimination, indistinguishible nodes, active variables in front, subscript compression,...

➤ Idea is old but first suggested by S. Eisenstat for speeding up sparse codes on vector machines.

➤ Beneficial on most machines

➤ Gains come in part from savings in Gather-Scatter operations.