Sparse Direct Solvers 1: The Challenge

Jennifer Scott
STFC Rutherford Appleton Laboratory
and the University of Reading

Woudschoten Conference, 3-5 October 2018

Overview

Aim to give you a brief flavour of what solving sparse linear systems using a direct method involves.

In some ways it is a complicated area (manipulating sparse data structures and writing robust and efficient software is a specialist skill).

But it involves very few theorems; instead it uses many heuristics, built upon years of experience developed by the small community of direct solver experts.

What is a sparse matrix?

Essentially its a matrix with a “small” number of non zero entries.

That is, a matrix that benefits from using special techniques to take advantage of the large number of zero entries.

The term was introduced by Harry Markowitz when describing the 1950s work on portfolio theory that won the 1990 Nobel Prize for Economics.

Note: any matrix not treated as sparse is regarded as dense

What is a direct method?

Involves an explicit factorization, such as Gaussian elimination

$$PAQ = LU$$

Permutations P and Q are chosen to preserve sparsity and maintain stability.
L: lower triangular, U: upper triangular

Solve

$$Ax = b$$

by solving two “easy” triangular systems

$$Ly = Pb$$

then

$$UQ^T x = y.$$

Just so we all remember ...

First step of Gaussian elimination (assume $a_{11} > 0$):

$$A = \begin{pmatrix} \alpha^2 & w^T \\ v & B \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ v/\alpha & I \end{pmatrix} \begin{pmatrix} \alpha & w^T \\ 0 & C \end{pmatrix}$$

$$C = B - vw^T/\alpha.$$

Now apply same process to C.

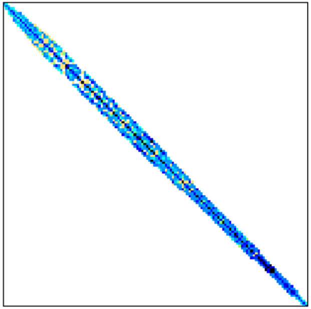
- First step of GE adds multiples of row 1 to later rows to eliminate subdiagonal entries in column 1.
- Second step adds multiples of (modified) row 2 to later rows to eliminate subdiagonal entries in column 2, and so on.
- At step k, multiples of (modified) row k are added to remaining rows $k+1$, ..., to obtain zeros below the diagonal in column k.
- Terminates after n steps where A is $n \times n$.

Where do we find sparse linear systems?

Oceanography	Oceans circulation,...
Meteorology	Climate change, weather forecasting,...
Structural mechanics	Elasticity, Plasticity, ...
Chemical engineering	Large molecules simulation,
Hydrology	Underground water remediation, pollution,..
Fluid-dynamics	Stokes, Navier-Stokes, advection diffusion,...
Economics	Econometric, linear programming, ...
Physics	Hamiltonian problems, thermodynamics,...
Biosciences	Ecology, diseases control,...
Computer science	Data mining, Googling, networks, ...
...	

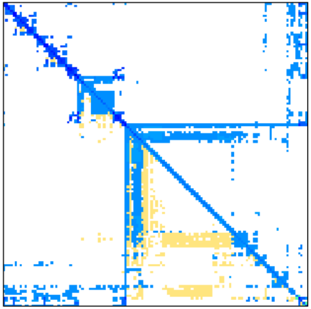
All give rise to matrices with different patterns and characteristics.

Structural engineering: Finite element model of a ship section



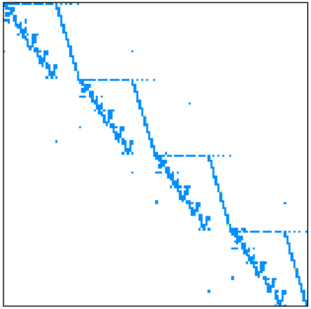
$n = 140,874$, $nz = 3,977,139$.
Symmetric structure, banded.

Circuit simulation problem



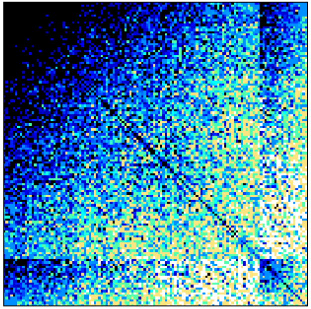
$n = 411,676$, $nz = 1,876,011$.
Unsymmetric sparsity pattern.

Chemical process engineering



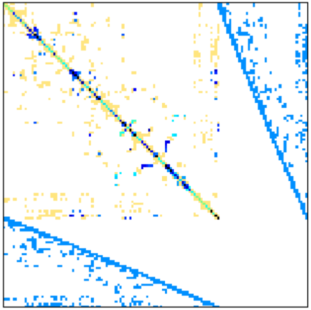
$n = 70,304$, $nz = 1,528,092$.
Highly unsymmetric pattern.

Social network problem: author collaborations



$n = 40,421$, $nz = 351,382$ (average entries per row ≈ 9).
Symmetric sparsity pattern.

Model of the Straz pod Ralskem mine



$n = 189,924,\ nz = 1,690,876.$

Symmetric indefinite matrix (note the zero block).

The last matrix is an example of a saddle-point matrix.

These have a block structure

$$\begin{pmatrix} A & B_1^T \\ B_2 & -C \end{pmatrix}.$$

In many applications,

- A is symmetric (and positive definite),
- $B_1 = B_2 = B$
- C is symmetric positive semi-definite (and often $C = 0$).

Such problems can present direct solvers with a number of challenges. Excellent survey paper (but mainly focusing on iterative methods) by Benzi, Golub, Liesen (2005).

Matrices and graphs

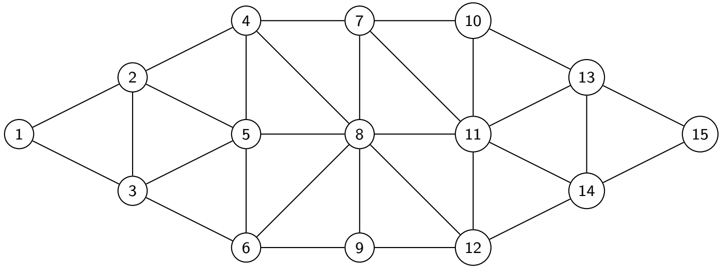
Key to the development of sparse direct methods is the relationship between matrices and graphs.

If the sparse symmetric matrix $A = \{a_{ij}\}$ is of order n, the *adjacency graph* $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$ of A is an undirected graph with *vertices* (or *nodes*) $\mathcal{V} = \{1, \ldots, n\}$ and edges $\mathcal{E}$, where an *edge* (u, v) is present in $\mathcal{E}$ if and only if $a_{uv} \neq 0$ $(u \neq v)$.

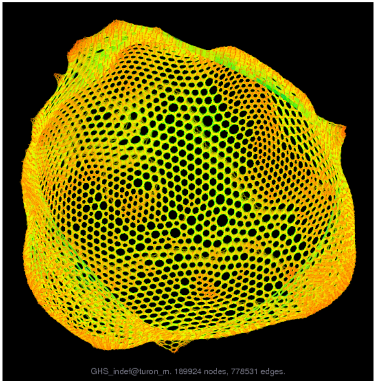
If A has an unsymmetric sparsity, a directed graph maybe used. Or the pattern of A^TA or $A + A^T$ is often used.

Correspondence between matrix A and its graph $\mathcal{G}(A)$

$$\begin{array}{r} 1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 7\\ 8\\ 9\\ 10\\ 11\\ 12\\ 13\\ 14\\ 15 \end{array}
\left(\begin{array}{ccccccccccccccc}
x & x & x & & & & & & & & & & & & \\
x & x & x & x & x & & & & & & & & & & \\
x & x & x & & x & x & & & & & & & & & \\
 & x & & x & x & & x & x & & & & & & & \\
 & x & x & x & x & x & & x & x & & & & & & \\
 & & x & & x & x & & & x & & & & & & \\
 & & & x & & & x & x & & x & x & & & & \\
 & & & x & x & & x & x & x & & x & x & & & \\
 & & & & x & x & & x & x & & & x & & & \\
 & & & & & & x & & & x & x & & x & & \\
 & & & & & & x & x & & x & x & x & x & x & \\
 & & & & & & & x & x & & x & x & & x & \\
 & & & & & & & & & x & x & & x & x & x \\
 & & & & & & & & & & x & x & x & x & x \\
 & & & & & & & & & & & & x & x & x
\end{array}\right)$$

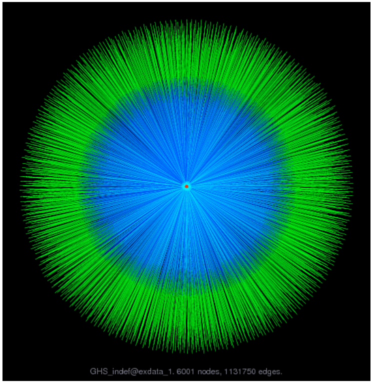


Model of the Straz pod Ralskem mine



Picture by Yifan Hu.

Optimization problem



What are the issues?

In a paper published in 1985, O'Connor explains why sparsity needs to be exploited to conserve memory and time when solving large linear systems:

- No computer available (in 1985) able to store in core a dense matrix of size 1000×1000.
- The microcomputers in common use (in 1985) typically have small memories.
- Linear system solvers are often a small but crucial part of a much larger algorithm. Typically, they can be used thousands of times in solving a single problem.

But aren't computers so large and powerful today that these issues have gone away?

How much memory do we need for a dense matrix?

2 GB of RAM holds a dense matrix of size $16,000 \times 16,000$.

A dense matrix of size $10^5 \times 10^5$ needs 75 GB of RAM.

A typical modern laptop now has 8 or 16 GB of RAM, while a workstation may have up to 64 GB.

Important: The operations needed to solve a dense system increase with the cube of the problem size.

There is a demand to solve modest-sized systems VERY FAST and also to solve EVER LARGER systems.

Why are sparse matrices more challenging than dense matrices?

- Difficulty is working/manipulating/keeping track of non zeros (and not destroying too many of the zeros).
- Need special data structures (including dynamic structures to allow non zero pattern to change during computation).
- Many operations on dense matrices use highly optimized routines (eg BLAS/LAPACK). Thus aim to design sparse algorithms using dense subproblems.
- Ratio of operations to data movement is small making parallelisation hard.

  Note: Processors have to communicate. The computational time may be reduced by adding more processors, but elapsed cost will depend on communications between processors.

A bit of history

1950s

- “Large” linear systems solved using relaxation methods and SOR or related techniques.
- Published papers were principally concerned with graph theory and combinatorics.

1960s

- The First Sparse Matrix Symposium was held at IBM Yorktown Heights in 1968 (124 participants).

  Interestingly, a number of current “hot” topics can be found in the proceedings of this meeting e.g.

  - Hybrid methods (combine direct and iterative procedures)
  - Matrix scaling
  - Use of single precision to get full accuracy
  - Permutations to block forms (used now for parallelism)

1970s

- Conference on Large Sparse Sets of Linear Equations held in summer of 1970 in Oxford.
- The first theses on sparse matrices written early '70s, notably Donald Rose (Harvard), Alan George (Stanford), Iain Duff (Oxford) and Andrew Sherman (Yale).
- Mid '70s first sparse matrix solvers:
  - Harwell Subroutine Library, including `MA28` (Duff '77)
  - YSMP (Yale)
  - SPARSPAK
- Late '70s onwards:
  - well-established field (number of small groups of experts)
  - available software increased (developed within companies such as IBM and Boeing as well as by academics)/books appeared/papers and reports/conferences etc
  - development never complete as architectures change plus an insatiable demand to solve ever faster and ever larger problems.

How large can we solve with direct method?

This has increased steadily. As a (very) rough indication:
1970: 200
1975: 1,000
1980: 10,000
1985: 100,000
1990: 250,000
1995: 500,000
2000: 2,000,000
2005: 10,000,000
2010: 1,000,000,000 ...
and ever LARGER

Increase been possible as computers became more powerful/more memory but also as a result of the years of effort that has gone into the development of algorithms and their efficient implementation.

Why do we want to use direct methods?

- They are robust.
- They can be used as black box solvers (user of backslash may have little understanding of what goes on behind the scenes).
- They can be easily incorporated into other software packages.
- They can produce reproducible results.
- When executed carefully, they can be extremely efficient for solving wide range of problems. It is often hard for an iterative solver to compete, except for very large problems

Let's illustrate how one sparse direct method proceeds

Assume A comes from a finite-element discretisation and so can be expressed as the sum of element matrices

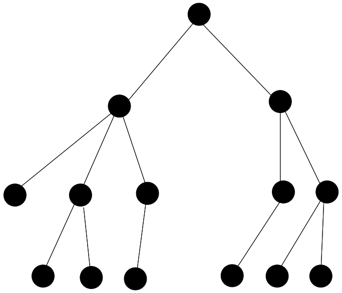
$$A = \sum_k A^{[k]}$$

where each $A^{[k]}$ is associated with a finite element and has nonzeros in only a few rows and columns corresponding to the variables in the element.

Basic multifrontal algorithm:

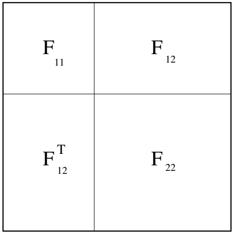
```
Given a pivot sequence (elimination order):
do for each pivot
    assemble all elements that contain the pivot into
    a (small) dense matrix;
    eliminate the pivot and any other variables that are
    found only here;
    treat the reduced matrix as a new generated element
end do
```

Assembly tree



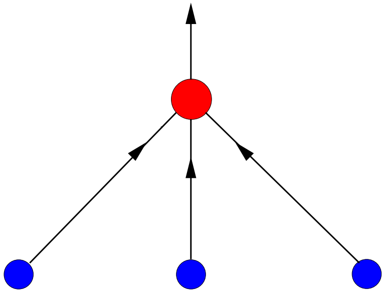
- Each leaf node represents an original element.
- Each non-leaf node represents set of eliminations and the corresponding generated element

At each non-leaf node: assemble the elements into a (small) dense matrix (called a frontal matrix).
After permutations, this has form:

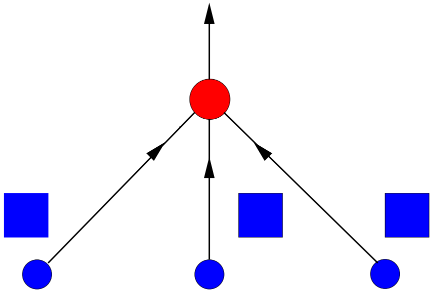


Pivot can only be chosen from F_{11} block since F_{22} is **NOT** fully summed. F_{11} is factorized then do Schur complement update.

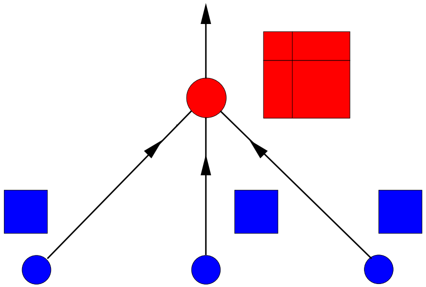
$$F_{22} \leftarrow F_{22} - F_{12}^T F_{11}^{-1} F_{12}$$



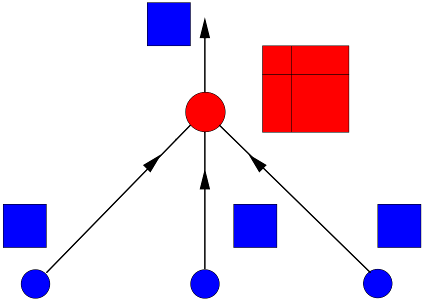
- Pass generated elements from children to parent



- Pass generated elements from children to parent
- At parent perform **ASSEMBLY** into dense frontal matrix



- Pass generated elements from children to parent
- At parent perform **ASSEMBLY** into dense frontal matrix
- Perform **ELIMINATIONS** using dense Gaussian elimination (allows Level 3 BLAS TRSM and GEMM)



- Pass generated elements from children to parent
- At parent perform **ASSEMBLY** into dense frontal matrix
- Perform **ELIMINATIONS** using dense Gaussian elimination (allows Level 3 BLAS TRSM and GEMM)
- Note: the computed entries of factor not needed again until the solve phase and so can be written to a file (out-of-core solver eg `HSL_MA77`)

Multifrontal methods

- Origin of method lies in frontal methods, developed initially by civil engineers for finite-element analysis (Irons 1970)
- Extended to a more general form allowing multiple fronts by Speelpenning (1973)
- Method very much associated with Duff and Reid (1983) (also George and Lui)
- Used in many sparse direct solvers (old and new) including `MA27` (1982), MUMPS, WSMP, `MA57`, `HSL_MA77` (out-of-core), `HSL_MA97`.
- Scope for parallelism (both within the tree and at the nodes)

What lies behind a direct solver?

We see from multifrontal outline that it comprises four steps:

1. Reorder the matrix (it matters what order you assemble the element matrices).
2. Analyse the pattern of the reordered matrix.
3. Scale and factorize the reordered matrix.
4. Solve the system(s) using the factors.

These are typical of all sparse direct solvers.

Many research years have been spent on each of these steps.

Observe that, in some cases, the cost of Steps 1 and 2 can be amortized over a series of problems.

Step 1: Reordering

Simple example:

$$(a) \quad \begin{array}{ccccc} x & x & x & x & x \\ x & x & & & \\ x & & x & & \\ x & & & x & \\ x & & & & x \end{array} \qquad\qquad (b) \quad \begin{array}{ccccc} x & & & & x \\ & x & & & x \\ & & x & & x \\ & & & x & x \\ x & x & x & x & x \end{array}$$

If we eliminate in order down the main diagonal then (a) fills in totally at step 1 (b) has no fill-in.

Step 1: Reordering

Let's focus on the case where A has a sparsity pattern that can be treated as symmetric ($a_{ij} \neq 0$ then $a_{ji} \neq 0$).

- How we label (number) the vertices in the adjacency graph $\mathcal{G}$ determines the sparsity pattern of the matrix A.
- Relabeling corresponds to permuting rows and columns of A.
- Why do this? We want to make the matrix "nice" before we try and solve the linear system.
- What is a good ordering depends on the choice of factorization algorithm but generally we want to:
  - limit flop counts and fill in the factors (maintain sparsity)
  - improve scope for parallelism. Can be conflicting aims.

Step 1: Reordering

Finding the optimal ordering is an NP-complete problem so heuristics used to find a"good" ordering.

An important class of ordering methods is based upon the minimum degree algorithm.

- First proposed as algorithm S2 by Tinney and Walker ('67).
- Uses a local strategy: at each stage of the factorization, the diagonal entry in the row of the remaining submatrix with the smallest number of entries is chosen as the next pivot candidate.
- Variants include Liu's ('85) multiple minimum degree algorithm and the approximate minimum degree (AMD) algorithm of Amestoy, Davis and Duff ('96) (designed to improve efficiency).
- Minimum degree is very good (and fast) for problems that are not too large (typically $n < 50,000$).

Nested dissection

An alternative is a global algorithm, such as nested dissection.

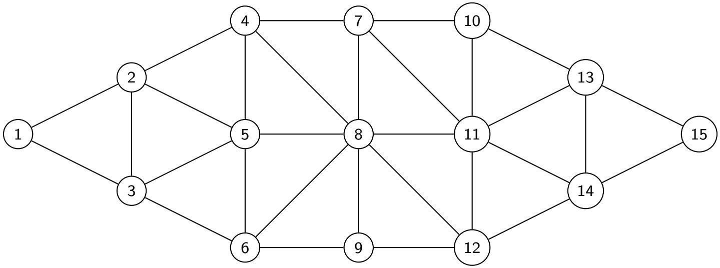
- First introduced by George ('73).
- Central concept: removal of a subset of vertices (called a separator $\mathcal{S}$) from $\mathcal{G}(\mathcal{V},\mathcal{E})$ that splits it into two disconnected parts $\mathcal{B}$ and $\mathcal{W}$.
- After reordering

$$A = \begin{pmatrix} A_{\mathcal{B}\mathcal{B}} & 0 & A_{\mathcal{B}\mathcal{S}} \\ 0 & A_{\mathcal{W}\mathcal{W}} & A_{\mathcal{W}\mathcal{S}} \\ A_{\mathcal{B}\mathcal{S}}^T & A_{\mathcal{W}\mathcal{S}}^T & A_{\mathcal{S}\mathcal{S}} \end{pmatrix}.$$

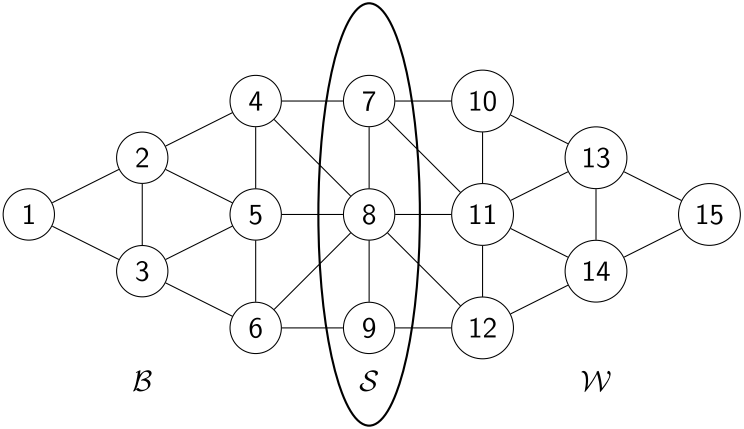
- Provided variables eliminated in the permuted order, no fill occurs within the zero blocks.
- Apply recursively to the sparse blocks $A_{\mathcal{B}\mathcal{B}}$ and $A_{\mathcal{W}\mathcal{W}}$.

Recall correspondence between matrix A and its graph $\mathcal{G}(A)$

$$\begin{array}{c}1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\end{array}\left(\begin{array}{ccccccccccccccc}
x & x & x & & & & & & & & & & & & \\
x & x & x & x & x & & & & & & & & & & \\
x & x & x & & x & x & & & & & & & & & \\
 & x & & x & x & & x & x & & & & & & & \\
 & x & x & x & x & x & & x & x & & & & & & \\
 & & x & & x & x & & & x & & & & & & \\
 & & & x & & & x & x & & x & x & & & & \\
 & & & x & x & & x & x & x & & x & x & & & \\
 & & & & x & x & & x & x & & & x & & & \\
 & & & & & & x & & & x & x & & x & & \\
 & & & & & & x & x & & x & x & x & x & x & \\
 & & & & & & & x & x & & x & x & & x & \\
 & & & & & & & & & x & x & & x & x & x \\
 & & & & & & & & & & x & x & x & x & x \\
 & & & & & & & & & & & & x & x & x
\end{array}\right)$$



Partitioned graph



Corresponding reordered matrix

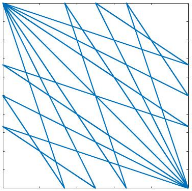
$$
\begin{array}{c}
1\\ 2\\ 3\\ 4\\ 5\\ 6\\ 10\\ 11\\ 12\\ 13\\ 14\\ 15\\ 7\\ 8\\ 9
\end{array}
\left(
\begin{array}{cccccc|cccccc|ccc}
x & x & x &   &   &   &   &   &   &   &   &   &   &   &   \\
x & x & x & x & x &   &   &   &   &   &   &   &   &   &   \\
x & x & x &   & x & x &   &   &   &   &   &   &   &   &   \\
  & x &   & x & x &   &   &   &   &   &   &   & x & x &   \\
  & x & x & x & x & x &   &   &   &   &   &   &   & x &   \\
  &   & x &   & x & x &   &   &   &   &   &   &   & x & x \\
\hline
  &   &   &   &   &   & x & x &   & x &   &   & x &   &   \\
  &   &   &   &   &   & x & x & x & x & x &   & x & x &   \\
  &   &   &   &   &   &   & x & x &   & x &   &   & x & x \\
  &   &   &   &   &   & x & x &   & x & x & x &   &   &   \\
  &   &   &   &   &   &   & x & x & x & x & x &   &   &   \\
  &   &   &   &   &   &   &   &   & x & x & x &   &   &   \\
\hline
  &   &   & x &   &   & x & x &   &   &   &   & x & x &   \\
  &   &   & x & x & x &   & x & x &   &   &   & x & x & x \\
  &   &   &   &   & x &   &   & x &   &   &   &   & x & x
\end{array}
\right)
$$

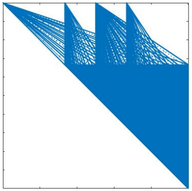
Nested dissection

- The quality depends crucially upon the choice of separators.
- Most widely-used software is MeTiS (Karypis and Kumar '95 onwards)
- ND is good for very large problems and problems arising from three-dimensional finite-element applications.
- Generally gives a more balanced elimination tree than minimum degree, which is beneficial for parallel direct solvers.

```
function $\pi$ = nested_dissection($A$, $PartitionAlg$)
Let $A$ have adjacency graph $\mathcal{G}(\mathcal{V},\mathcal{E})$
if  dissection has terminated  then
    $\pi = \mathtt{AMD}(\mathcal{V},\mathcal{E})$
else
    $\phi(\mathcal{B},\mathcal{W},\mathcal{S}) = PartitionAlg(\mathcal{V},\mathcal{E})$
    $\pi_{\mathcal{B}} = \mathtt{nested\_dissection}(A_{\mathcal{B}\mathcal{B}}, \mathtt{PartitionAlg})$
    $\pi_{\mathcal{W}} = \mathtt{nested\_dissection}(A_{\mathcal{W}\mathcal{W}}, \mathtt{PartitionAlg})$
    $\pi_{\mathcal{S}}$ is an ordering of $\mathcal{S}$
    $\pi = \begin{bmatrix} \pi_{\mathcal{B}} \\ \pi_{\mathcal{W}} \\ \pi_{\mathcal{S}} \end{bmatrix}$
end if
```

Optimization problem

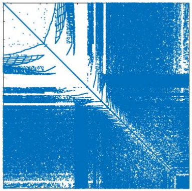


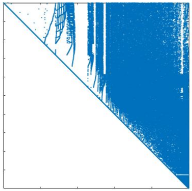


Original matrix A (left): $n = 50,000$, $nz = 349,968$

Cholesky factor $U = L^T$ (right): $nz = 231,509,103$ (ratio $= 661$)

Optimization problem

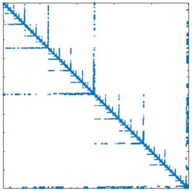


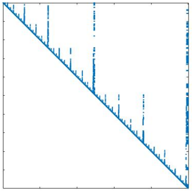


Original matrix after AMD ordering (left): $nz = 349,968$

Factor of reordered matrix (right): $nz = 4,331,858$ (ratio = 12.4)

Optimization problem





Original matrix after ND ordering (left): $nz = 349,968$

Factor of reordered matrix (right): $nz = 2,092,145$ (ratio $= 5.98$)

Summary ... a few observations

Sparse Direct Methods have had a relatively short, distinguished, and very active history.

Sparse Direct Methods are complicated and require a degree of expertise and, as we shall see from the other lectures, sophisticated programming skills.

And while Sparse Direct Methods are VERY widely used, the active sparse direct solver research community is rather small, with a limited number of expert groups.

In my second lecture, we will look at the other stages after reordering ...