Sparse Matrices
Introduction to sparse matrices and direct methods

Iain Duff

STFC Rutherford Appleton Laboratory and CERFACS

Summer School
The 6th de Brùn Workshop. Linear Algebra and Matrix Theory: connections, applications and computations. NUI Galway, Ireland. 3-7 December 2012.

Lecture 1

- Introduction to sparse matrices
- Introduction to graphs and matrices
- Introduction to solution of sparse equations
- Introduction to direct methods

Wish to solve

$$\mathbf{Ax} = \mathbf{b}$$

where **A** is

LARGE

and

S P A R S E

By LARGE we mean matrices of large order n. This is a function of time.

t	n(t)
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

The meaning of sparse is not so simple

SPARSE	...	NUMBER ENTRIES
kn		$k \sim 2 - \log n$

NUMERICAL APPLICATIONS

Stiff ODEs ... BDF ... Sparse Jacobian

Linear Programming
..... simplex
..... interior point

Optimization/Nonlinear Equations

Elliptic Partial Differential equations

Eigensystem Solution

Two Point Boundary Value Problems

Least Squares Calculations

APPLICATION AREAS

Physics	CFD
	Lattice gauge
	Atomic spectra
Chemistry	Quantum chemistry
	Chemical engineering
Civil engineering	Structural analysis
Electrical engineering	Power systems
	Circuit simulation
	Device simulation
Geography	Geodesy
Demography	Migration
Economics	Economic modelling
Behavioural sciences	Industrial relations
Politics	Trading
Psychology	Social dominance
Business administration	Bureaucracy
Operations research	Linear Programming

Sparse matrices

THERE FOLLOWS PICTURES OF SPARSE MATRICES FROM VARIOUS APPLICATIONS

This is done to illustrate different structures for sparse matrices

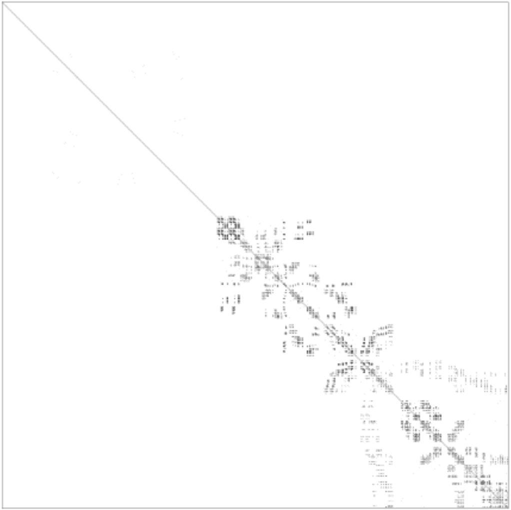
Sparse matrices

Thermal Simulation; SHERMAN2

Sparse matrices

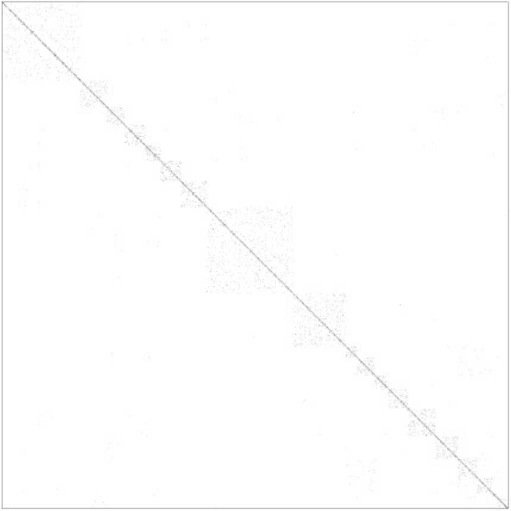
Weather Matrix; FS 760 3

Sparse matrices



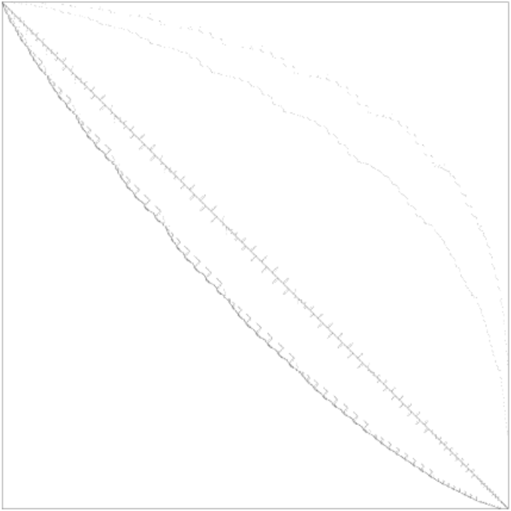
Dynamic Calculation in Structures; BCSSTM13

Sparse matrices



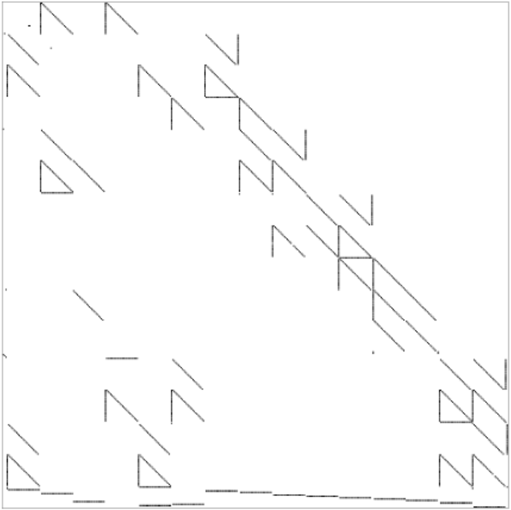
Power Systems; BCSPWR07

Sparse matrices



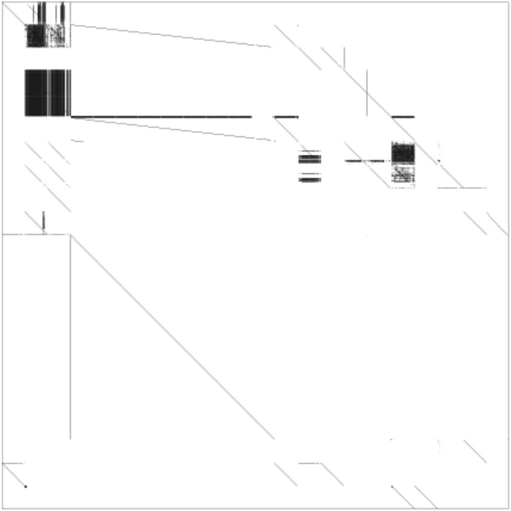
Simulation of Computing Systems; GRE 1107

Sparse matrices



Chemical Engineering; WEST0381

Sparse matrices



Economic Modelling; ORANI678

STANDARD SETS OF SPARSE MATRICES

Original set

Harwell-Boeing Sparse Matrix Collection

Extended set of test matrices available from:

`http://www.cise.ufl.edu/research/sparse/matrices`

and

Matrix market

`http://math.nist.gov/MatrixMarket`

Large and increasing collection maintained by the GRID-TLSE Project

`http://gridtlse.org/`

Matrix storage schemes

For efficient solution of sparse equations we must

- Only store nonzeros (or exceptionally a few zeros also)
- Only perform arithmetic with nonzeros
- Preserve sparsity during computation

DATA STRUCTURES FOR SPARSE MATRICES

Here are a few data structures used for storing sparse matrices. The best scheme is very dependent on the structure of the matrix and the way in which sparsity is to be exploited.

COORDINATE SCHEME The matrix is held as a collection of triplets (i, j, a_{ij}) where the entry (i, j) of the matrix has value a_{ij}. This is used by the Matrix Market and MATLAB.

CSR (CSC) In the compressed sparse row (or equivalently column) scheme, the matrix is held as a collection of sparse vectors, one for each row (or column). Entries in a vector are held as the pair (i, a_i) where the ith component of the vector has value a_i.

LINKED LIST With each entry we hold one or more links to other entries. Typically the row (and/or column) of the matrix can be recovered by running through the links.

Sparse data structures

STRUCTURED STORAGE The matrix may be held by diagonals or, for each row, all entries from the first nonzero to the diagonal are stored. These schemes will normally store explicit zeros but can be efficient for particular structures.

ELEMENTAL Matrix is represented as an expanded sum $A = \sum_k A^{[k]}$, where each $A^{[k]}$ is held as a dense matrix.

HASH CODING A map is generated from $I^n \times I^n$ to $[1, nz]$ with procedures for handling collisions (since $nz \ll n^2$).

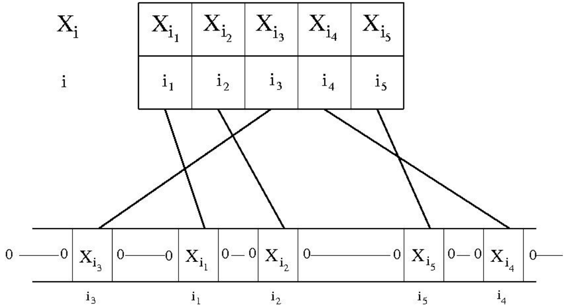
BIT MAPS A Boolean map indicates the positions of nonzero entries in the matrix.

Sparse data structures

SPARSE VECTOR STORAGE

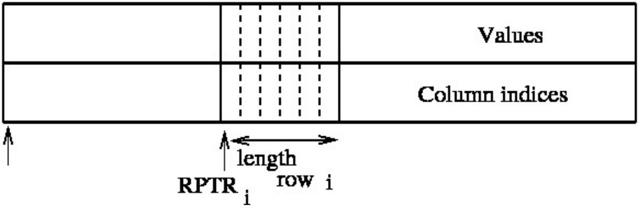
$$\mathbf{X} = (\mathbf{x_1}\ \mathbf{x_2}\ \ldots \mathbf{x_n})^{\mathbf{T}}$$

For each i such that $x_i \neq 0$, store i and x_i

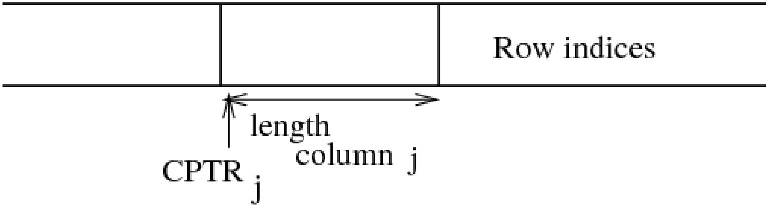


GATHER/SCATTER PACK/UNPACK

CSR (Compressed Sparse Row)

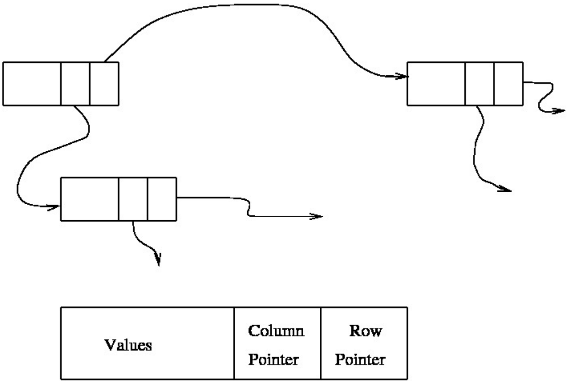


Usually with separate copy for access to nonzero pattern by columns:



Sparse data structures

Linked lists



Sparse data structures

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

A 5×5 sparse matrix.

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.

Table: The matrix stored in the coordinate scheme.

Sparse data structures

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.

Table: Matrix stored as a collection of sparse row vectors.

Subscripts	1	2	3	4	5	6	7	8	9	10	11
IROWST	4	3	6	10	11						
JCN	4	5	1	1	5	2	4	3	3	2	1
VAL	-1.	3.	2.	1.	6.	-3.	-4.	-5.	-2.	4.	5.
LINK	0	0	9	1	0	0	0	5	2	7	8

Table: The matrix held as a linked list.

Graphs and matrices

A graph $G(V, E)$ is a set of vertices (or nodes), V, and a set of edges, E where an edge (v_i, v_j) of E is a pair of vertices v_i and v_j of V.
Although in sparse matrix research we use many different graphs, we will mainly associate three types of graphs with sparse matrices viz.

- An (undirected) graph on on n vertices can be associated with a symmetric matrix of order n. Edge (i, j) exists in the graph if and only if entry a_{ij} (and, by symmetry a_{ji}) in the matrix is nonzero.
- A directed graph on n vertices can be associated with a matrix of order n. Edge (i, j) exists in the graph if and only if entry a_{ij} in the matrix is nonzero.

Graphs and matrices

- A bipartite graph $G_B = (V_r, V_c, E)$ consists of two disjoint vertex sets V_r and V_c and an edge set E. The sets V_r and V_c correspond to rows and columns of the sparse matrix respectively so that an edge from veretex v_i of V_r to vertex v_j of V_c exists if and only if entry (v_i, v_j) of the matrix is nonzero. Note that the cardinality of sets V_r and V_c need not be the same so that this representation can be used for rectangular matrices.

Graphs and sparse matrices

Main benefits of using graphs to represent sparse matrices

- Structure of graph is invariant under symmetric permutations of the matrix (corresponds to relabelling of vertices).
- For mesh problems, there is usually an equivalence between the mesh and the graph associated with the resulting matrix. We thus work directly with the underlying structure.
- We can represent cliques in graphs by listing vertices in a clique without storing all the interconnecting edges.

Graphs and matrices

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

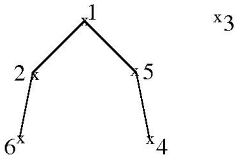
A symmetric matrix.

Graphs and matrices

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

A symmetric matrix.

has graph



Graphs and sparse matrices

Hence with ordering
[3 4 5 1 2 6]
the resulting symmetrically permuted matrix has the pattern:

Graphs and sparse matrices

Hence with ordering
[3 4 5 1 2 6]
the resulting symmetrically permuted matrix has the pattern:

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

Reordered symmetric matrix.

Graphs and sparse matrices

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

An unsymmetric matrix

Graphs and sparse matrices

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

An unsymmetric matrix

Has graph

Graphs and sparse matrices

Reorder symmetrically ......
4 5 1 2 3 6

$$
\begin{array}{ccc|ccc}
\times & \times & & & & \\
 & \times & \times & & & \\
\times & & \times & \times & & \\
\hline
 & & & \times & \times & \\
 & & & & \times & \times \\
 & & & \times & & \times
\end{array}
$$

HSL

- Formerly Harwell Subroutine Library
- Started at Harwell in **1963**
- Most routines from research work of group
- Particular strengths in:
  - Sparse Matrices
  - Optimization (also GALAHAD)
  - Large-scale system solution
- Last main release was 2011

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

GALAHAD:
`http://www.galahad.rl.ac.uk`

MATLAB

help sparfun

Commands:
nnz(), issparse()

Best command:
spy()

Some matrices

- gallery('wathen',20,15)
- bucky
- load west0479
- delsq(numgrid('S',30))
- speye()

See demos

MATLAB is continually improving its coverage of sparse matrices and sparse matrix operations including solvers

Many HSL codes accessible through MATLAB

Solution of linear systems

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater.

Solution of linear systems

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater.

There are two main techniques viz.

Solution of linear systems

We wish to solve the linear system

$$\mathbf{Ax} \ = \ \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater.

There are two main techniques viz.

- Direct methods (based on matrix factorization)

Solution of linear systems

We wish to solve the linear system

$$\mathbf{Ax} = \mathbf{b}$$

where the sparse matrix $\mathbf{A}$ has dimension 10^6 or greater.

There are two main techniques viz.

- Direct methods (based on matrix factorization)
- Iterative methods (with some form of preconditioning)

Solving sparse systems

$$\mathbf{Ax} = \mathbf{b}$$

has solution

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

Solving sparse systems

$$\mathbf{Ax} = \mathbf{b}$$

has solution

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

BUT

This is notational only.

Solving sparse systems

$$\mathbf{Ax} = \mathbf{b}$$

has solution

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

BUT

This is notational only.

Do not use or even think of using inverse of **A**.

Solving sparse systems

$$\mathbf{Ax} = \mathbf{b}$$

has solution

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

BUT

This is notational only.

Do not use or even think of using inverse of $\mathbf{A}$.

For sparse $\mathbf{A}$,

$$\mathbf{A}^{-1} \text{ is usually dense.}$$

Examples are:

Tridiagonal and arrowhead

DIRECT METHODS

Gaussian Elimination

$$\mathbf{PAQ} \rightarrow \mathbf{LU}$$

Permutations $\mathbf{P}$ and $\mathbf{Q}$ chosen to preserve sparsity and maintain stability

$\mathbf{L}$: Lower triangular (sparse)
$\mathbf{U}$: Upper triangular (sparse)

We then solve:

$$\mathbf{Ax} = \mathbf{b}$$

by

$$\mathbf{Ly} = \mathbf{Pb}$$

then

$$\mathbf{UQ^T x} = \mathbf{y}$$

Complexity of LU factorization on matrices of order n

Full (dense) matrix	$\frac{2}{3}n^3 + \mathcal{O}(n^2)$	flops
	n^2	storage

For band matrix (order n, bandwidth k)

$2k^2n$ work, nk storage

Five-diagonal matrix (on $k \times k$ grid)

$\mathcal{O}(k^3)$ work
and
$\mathcal{O}(k^2 \log k)$ storage

Tridiagonal + Arrowhead matrix

$\mathcal{O}(n)$ work and storage

Target $\mathcal{O}(n) + \mathcal{O}(\tau)$ for sparse matrix of order n with τ entries.

Complexity of sparse direct method in 2 and 3 dimensions

Grid dimensions	Matrix order	Work to factorize	Factor storage
$k \times k$	k^2	k^3	$k^2 \log k$
$k \times k \times k$	k^3	k^6	k^4

$\mathcal{O}$ complexity of direct method on 2D and 3D grids.

An $\mathcal{O}(n)$ method!!

For problems where there is a natural decay of entries in the inverse, then the use of low rank approximations coupled with suitable orderings can provably result in a method that scales linearly with problem size.

Direct methods

To indicate how powerful parallel direct methods are, we note that the PARASOL test problem

AUDIKW_1

of order

943,695

with

39.3 million

entries
is now a standard test/training problem.

DIRECT METHODS

- Easy to package
- High accuracy
- Method of choice in many applications
- Not dramatically affected by conditioning
- Reasonably independent of structure

However

- High time and storage requirement
- Typically limited to $n \sim 1000000$

DIRECT METHODS

- Easy to package
- High accuracy
- Method of choice in many applications
- Not dramatically affected by conditioning
- Reasonably independent of structure

However

- High time and storage requirement
- Typically limited to $n \sim 1000000$

So

DIRECT METHODS

- Easy to package
- High accuracy
- Method of choice in many applications
- Not dramatically affected by conditioning
- Reasonably independent of structure

However

- High time and storage requirement
- Typically limited to $n \sim 1000000$

So

- Use on subproblem
- Use as preconditioner

Phases of sparse direct solution

Although the exact subdivision of tasks for sparse direct solution will depend on the algorithm and software being used, a common subdivision is given by:

ANALYSE An analysis phase where the matrix structure is analysed to produce a suitable ordering and data structures for efficient factorization.

FACTORIZE A factorization phase where the numerical factorization is performed.

SOLVE A solve phase where the factors are used to solve the system using forward and backward substitution.

Phases of sparse direct solution

We note the following:

- ANALYSE is sometimes preceded by a preordering phase to exploit structure.
- For general unsymmetric systems, the ANALYSE and FACTORIZE phases are sometimes combined to ensure the ordering does not compromise stability.
- Note that the concept of separate ANALYSE and FACTORIZE phases is not present for dense systems.

Steps Match Solution Requirements

1. One-off solution

$$Ax = b \qquad\qquad \text{A/F/S}$$

2. Sequence of solutions [Matrix changes but structure is invariant]

$$\begin{aligned} A_1 x_1 &= b_1 \\ A_2 x_2 &= b_2 \\ A_3 x_3 &= b_3 \end{aligned} \qquad\qquad \text{A/F/S/F/S/F/S}$$

3. Sequence of solutions [Same matrix]

$$\begin{aligned} A x_1 &= b_1 \\ A x_2 &= b_2 \\ A x_3 &= b_3 \end{aligned} \qquad\qquad \text{A/F/S/S/S}$$

For example ....

2. Solution of nonlinear system [A is Jacobian]
3. Inverse iteration

TYPICAL RATIO OF TIMES

ANALYSE/FACTORIZE	100
FACTORIZE	10
SOLVE	1

The actual ratio in any instance will depend on the matrix, the computer, and the code used but the above ratio is very typical

The ordering problem

In using Gaussian elimination to solve

$$\mathbf{Ax} = \mathbf{b},$$

we perform the decomposition

$$\mathbf{PAQ} = \mathbf{LU}$$

where

$\mathbf{P}$ and $\mathbf{Q}$ ($\mathbf{P}^\top$ if $\mathbf{A}$ is symmetric) are permutation matrices chosen to:

1. preserve sparsity and reduce work in decomposition and solution
2. enable use of efficient data structures
3. ensure stability
4. take advantage of structure

Ordering

Benefits of Sparsity Ordering

Matrix of Order 2021 with 7353 entries

Procedure	Total storage (Kwords)	Flops
Treating system as dense	4084	5503
Storing and operating only on nonzero entries	71	1073
Using sparsity pivoting	14	42

Ordering

Two categories ...

LOCAL Ordering

and

GLOBAL Ordering

Ordering for symmetric matrices

$$
\begin{array}{ccccc}
\times & \times & \times & \times & \times \\
\times & \times & & & \\
\times & & \times & & \\
\times & & & \times & \\
\times & & & & \times
\end{array}
\quad \xrightarrow{L/U} \quad
\begin{array}{ccccc}
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times
\end{array}
$$

$$
\begin{array}{ccccc}
\times & & & & \times \\
 & \times & & & \times \\
 & & \times & & \times \\
 & & & \times & \times \\
\times & \times & \times & \times & \times
\end{array}
\quad \xrightarrow{L/U} \quad
\begin{array}{ccccc}
\times & & & & \times \\
 & \times & & & \times \\
 & & \times & & \times \\
 & & & \times & \times \\
\times & \times & \times & \times & \times
\end{array}
$$

Minimum degree (Tinney S2 ... 1967)

- At each stage choose diagonal entry with least number of entries in row of reduced matrix.
- Using this to minimize the fill-in is NP-complete
- Usually it is a good heuristic and gives a good ordering.
- However, being a local algorithm, it does not take full account of the global underlying structure of the problem.
- Its performance can be unpredictable and it is sensitive to how "ties" are broken.
- The code for efficient implementation can be remarkably complicated.
- Often the most time consuming part of the algorithm is the degree update $\longrightarrow$ Approximate Minimum Degree (AMD).

Local ordering for unsymmetric matrices

$$\begin{matrix} * & \times & \times \\ \times & v & v \\ \times & v & v \\ \times & v & v \end{matrix}$$

$$r_i = 3,\ c_j = 4$$

$$\text{Minimize } (r_i - 1) * (c_j - 1)$$

MARKOWITZ ORDERING (Markowitz 1957)

Choose nonzero satisfying a numerical criterion with best Markowitz count.

Threshold pivoting for stability

a_{ij} is admissible as a pivot if

$$|a_{ij}| \geq u \max_k |a_{kj}| \qquad 0 < u \leq 1$$

So choose as pivot at each stage the entry with lowest Markowitz cost satisfying the above inequality.

Called threshold pivoting

Threshold pivoting for stability

Growth at one step is bounded by

$$\max_i |a_{ij}^{(k+1)}| \leq (1 + 1/u) \max_i |a_{ij}^{(k)}|$$

and overall growth by

$$\max_i |a_{ij}^{(k)}| \leq (1 + 1/u)^{p_j} \max_i |a_{ij}|$$

where p_j is the number of off-diagonal entries in the jth column of U.

If

$$\mathbf{H} = \hat{\mathbf{L}}\hat{\mathbf{U}} - \mathbf{A}$$

then

$$|h_{ij}| \leq 5.01 \epsilon n \max_k |a_{ij}^{(k)}|$$

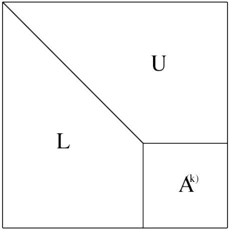
This means that through u, the threshold parameter, we can directly influence backward error in factorization.

Threshold pivoting for stability

Threshold	Entries in factors	Error in solution
1.0	16767	3×10^{-9}
0.25	14249	6×10^{-10}
0.10	13660	4×10^{-9}
0.01	15045	1×10^{-5}
10^{-4}	16198	1×10^{2}
10^{-10}	16553	3×10^{23}

Effect of changing threshold parameter. Matrix is order 541 with 4285 entries.

Efficient implementation of Markowitz



If you search all of $A^{(k)}$ each time, the complexity is $\mathcal{O}(n\tau) \sim \mathcal{O}(n^2)$.

Want $\mathcal{O}(1)$ search every time ....

Search rows/columns in order of increasing sparsity

Can limit search to small number of rows/columns

Global ordering for symmetric matrices

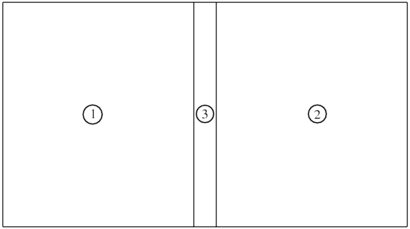
PARTITIONING

Divide and conquer paradigm

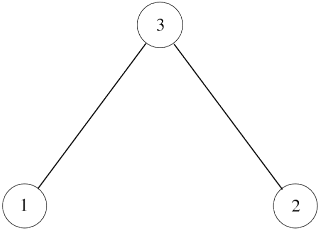
(Nested) dissection

Domain decomposition

Substructuring



Global ordering for symmetric matrices



Computational Graph

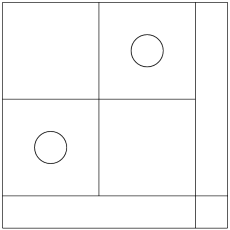
Global ordering for symmetric matrices

Dissection

Chooses last pivots first

Dissect problem into two parts

Resulting matrix has form:



Hence to nested dissection

Nested dissection

- Nested dissection was originally proposed by Alan George in his thesis in 1971.
- Although good on regular grids and for complexity was not used for general symmetric matrices until 1990s
- Key to achieving good ordering on general problems is **Bisection Algorithm**

Comparison of AMD and Nested Dissection

A nested dissection ordering can be generated by the graph partitioning software called METIS (Karypis and Kumar) or SCOTCH (Pellegrini)

For large problems this often beats a local ordering. For example, AMD (Amestoy, Davis, and Duff)

Matrices are from PARASOL test set and runs are from a very early version of MUMPS (more about this later).

	Entries in factors (10^6)	Operations in factorization (10^9)
MIXTANK		
AMD	38.5	64.4
ND	18.9	13.2
INVEXTR1		
AMD	30.3	35.8
ND	15.7	8.1
BBMAT		
AMD	46.0	41.6
ND	35.7	25.7

MA48

An example of a code that uses Markowitz and Threshold Pivoting
is

the HSL Code

`MA48`

Features of MA48

- Uses Markowitz with threshold pivoting
- Factorizes rectangular and singular systems
- Block Triangularization
  - Done at "higher" level
  - Internal routines work only on single block
- Switch to full code at all phases of solution
- Three factorize entries
  - Only pivot order is input
  - Fast factorize .. uses structures generated in first factorize
  - Only factorizes later columns of matrix
- Iterative refinement and error estimator in Solve phase
- Can employ drop tolerance
- "Low" storage in analyse phase

Direct codes

TYPICAL INNERMOST LOOP

```
DO 590 JJ = J1, J2
     J = ICN (JJ)
     IF   (IQ(J). GT.0)   GO TO 590
     IOP = IOP + 1
     PIVROW = IJPOS - IQ (J)
     A(JJ) = A(JJ) + AU * A (PIVROW)
     IF (LBIG)  BIG = DMAXI (DABS(A(JJ)), BIG)
     IF (DABS(A(JJ)). LT. TOL) IDROP = IDROP + 1
     ICN (PIVROW) =      ICN (PIVROW)
590 CONTINUE
```

MA48 .. why look at other codes

Why look at other codes??

- MA48 does not naturally vectorize or parallelize
  - short vector lengths
- There is heavy use of indirect addressing in MA48 (viz. references of form W(IND(I))
  - Even with h/ware indirect addressing is 2-4 times slower
  - more memory traffic
  - non-localization of data
- MA48 can perform poorly when there is significant fill-in
- We would like to take more advantage of the Level 3 BLAS