Euro-Par '99

TUTORIAL
ON
LINEAR ALGEGRA ALGORITHMS
AND SOFTWARE
FOR
LARGE SCIENTIFIC PROBLEMS

Sparse Direct methods
and software for
systems of equations

Iain S Duff
Rutherford Appleton Laboratory
CERFACS
Strathclyde University
http://www.cse.clrc.ac.uk/Person/I.S.Duff

Wish to solve

$$Ax = b$$

A is

LARGE

and

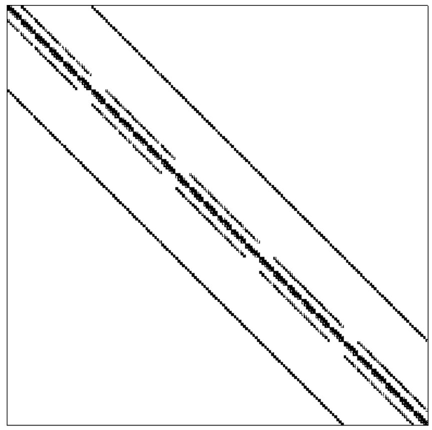
S P A R S E

APPLICATION AREAS

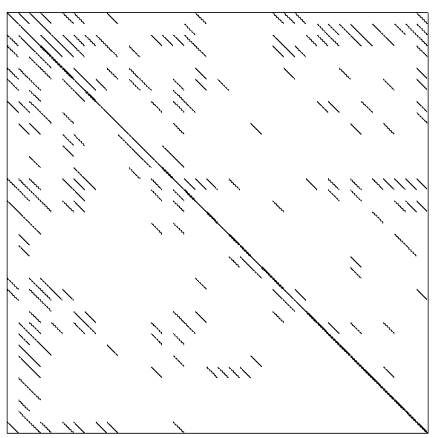
Physics	**CFD**
	Lattice gauge
	Atomic spectra
Chemistry	**Quantum chemistry**
	Chemical engineering
Civil Engineering	**Structural analysis**
Electrical Engineering	**Power systems**
	Circuit simulation
	Device simulation
Geography	**Geodesy**
Demography	**Migration**
Economics	**Economic modelling**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

THERE FOLLOWS PICTURES OF SPARSE MATRICES FROM VARIOUS APPLICATIONS

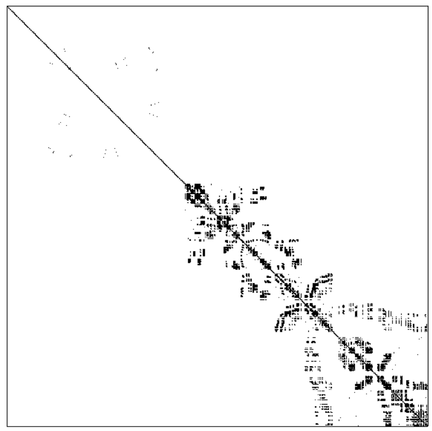
This is done to illustrate different structures for sparse matrices



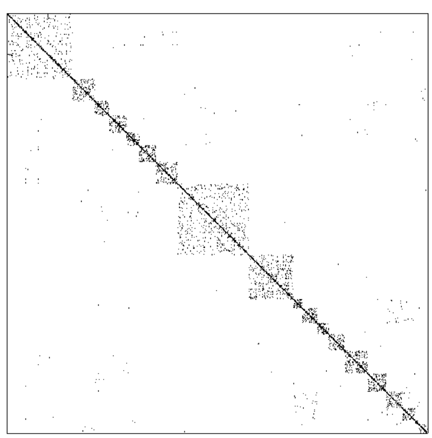
Thermal Simulation; SHERMAN2



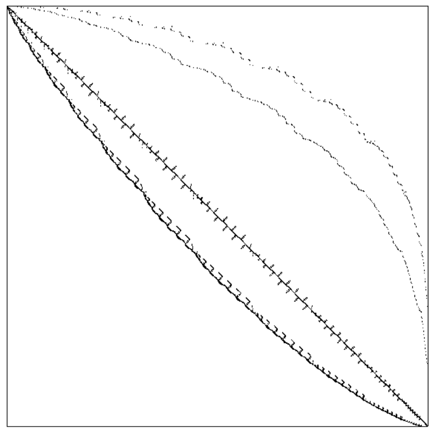
Weather Matrix; FS 760 3



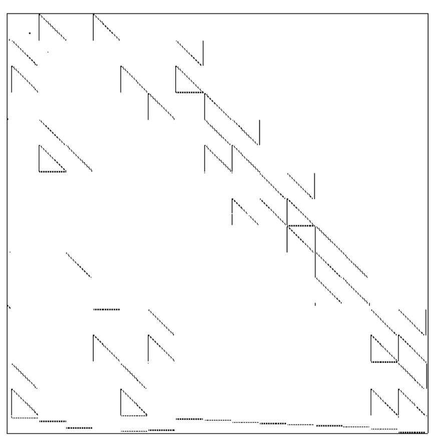
Dynamic Calculation in Structures; BCSSTM13



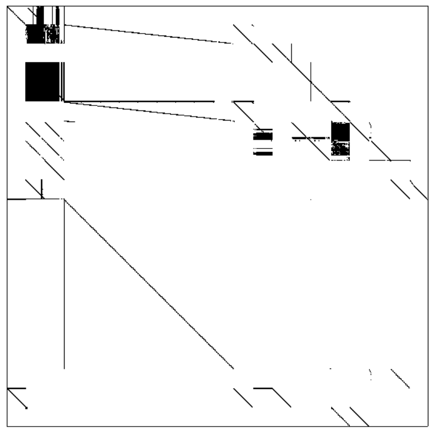
Power Systems; BCSPWR07



Simulation of Computing Systems; GRE 1107



Chemical Engineering; WEST0381



Economic Modelling; ORANI678

STANDARD SETS OF SPARSE MATRICES

Original set

Harwell-Boeing Sparse Matrix Collection

Available by anonymous ftp from:
ftp.numerical.rl.ac.uk

or from the Web page:
http://www.cse.clrc.ac.uk/Activity/SparseMatrices

Extended set of test matrices available from:

Tim Davis
ftp.cise.ufl.edu
cd pub/faculty/davis/matrices

RUTHERFORD-BOEING SPARSE MATRIX TEST COLLECTION

- **Unassembled finite-element matrices**
- **Least-squares problems**
- **Unsymmetric eigenproblems**
- **Large unsymmetric matrices**
- **Very large symmetric matrices**
- **Singular Systems**

See Report:

The Rutherford-Boeing Sparse Matrix Collection
Iain S Duff, Roger G Grimes and John G Lewis
Report RAL-TR-97-031, Rutherford Appleton Laboratory

http://www.numerical.rl.ac.uk/reports/reports

To be distributed via matrix market

http://math.nist.gov/MatrixMarket

DIRECT METHODS

Gaussian Elimination

$$PAQ \rightarrow LU$$

Permutations P and Q chosen to preserve sparsity and maintain stability

L : Lower triangular (sparse)
U : Upper triangular (sparse)

SOLVE:

$$Ax = b$$

by

$$Ly = Pb$$

then

$$Uqx = y$$

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 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.**

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 a separate FACTORIZE phase is not present for dense systems.**

STEPS MATCH SOLUTION REQUIREMENTS

1. One-off solution

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b} \qquad \text{A/F/S}$$

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

$$\mathbf{A}_1\boldsymbol{x}_1 = \boldsymbol{b}_1$$
$$\mathbf{A}_2\boldsymbol{x}_2 = \boldsymbol{b}_2 \qquad \text{A/F/S/F/S/F/S}$$
$$\mathbf{A}_3\boldsymbol{x}_3 = \boldsymbol{b}_3$$

3. Sequence of solutions
[Same matrix]

$$\mathbf{A}\boldsymbol{x}_1 = \boldsymbol{b}_1$$
$$\mathbf{A}\boldsymbol{x}_2 = \boldsymbol{b}_2 \qquad \text{A/F/S/S/S}$$
$$\mathbf{A}\boldsymbol{x}_3 = \boldsymbol{b}_3$$

For example ….

2. Solution of nonlinear system

A is Jacobian

3. Inverse iteration

TIMES (MA48) FOR 'TYPICAL' EXAMPLE

[Seconds on CRAY YMP … matrix GRE 1107]

ANALYSE/FACTORIZE	**.66**
FACTORIZE	**.075**
SOLVE	**.0068**

DIRECT METHODS

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

However

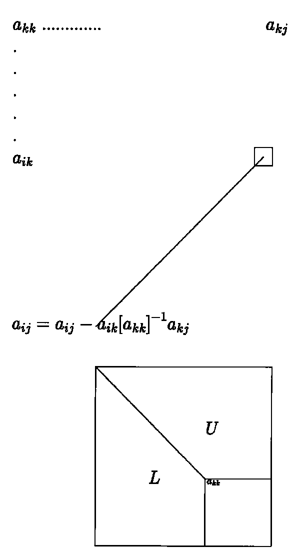
- **High time and storage requirement**
- **Typically limited to $n \sim 100000$**

So

- **Use on subproblem**
- **Use as preconditioner**

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**



Fill-in in Gaussian Elimination

Ordering for sparsity

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

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

MINIMUM DEGREE (Tinney S2 ... 1967)

At each stage choose diagonal entry with least number of entries in row of reduced matrix.

Ordering for sparsity … unsymmetric matrices

$$\begin{array}{ccc} * & x & x \\ x & v & v \\ x & v & v \\ x & v & v \end{array}$$

$r_i = 3, c_j = 4$

Minimize $(r_i - 1) * (c_j - 1)$

MARKOWITZ ORDERING (Markowitz 1957)

Choose nonzero satisfying numerical criterion with best Markowitz count

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*

Benefits of Sparsity on Matrix of Order 2021 with 7353 entries

Procedure	Total Storage (Kwords)	Flops (10^6)	Time (secs) CRAY J90
Treating system as dense	4084	5503	34.5
Storing and operating only on nonzero entries	71	1073	3.4
Using sparsity pivoting	14	42	0.9

HARWELL SUBROUTINE LIBRARY

- Started at Harwell in 1963
- Most routines from research work of group
- Particular strengths in:
  - Sparse Matrices
  - Optimization
  - Large-Scale system solution
- Since 1990 Collaboration between RAL and Harwell

RELEASE 12 … January 1996

URL: http://www.cse.clrc.ac.uk/Activity/HSL

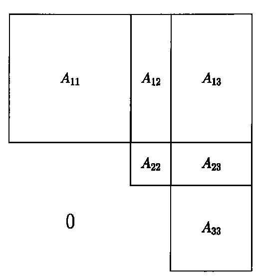
Wide Range of Sparse Direct Software

• Markowitz + Threshold	MA48
• Band - Variable Band (Skyline)	MA55
• Frontal	
- Symmetric	MA62
- Unsymmetric	MA42
• Multifrontal	
- Symmetric	MA27 and MA47
- Unsymmetric	MA41
- Very Unsymmetric	MA38
- Unsymmetric element	MA46
• Supernodal	SUPERLU

Features of MA48

- **Uses Markowitz with threshold privoting**
- **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**

BLOCK TRIANGULAR FORM



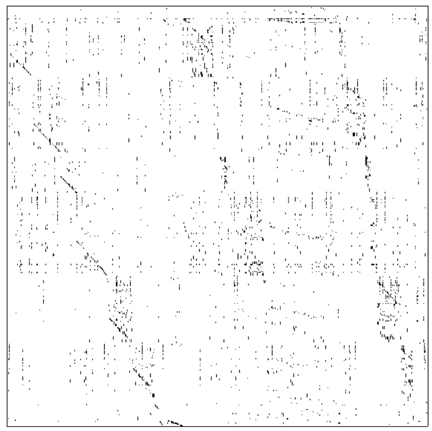
The overall system is solved through the steps ....

$$A_{33}X_3 = B_3$$

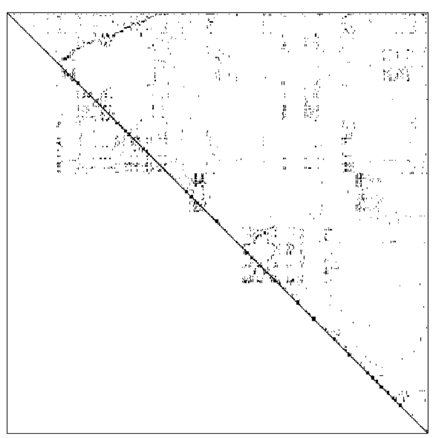
$$A_{22}X_2 = B_2 - A_{23}X_3$$

$$A_{11}X_1 = B_1 - A_{13}X_3 - A_{23}X_3$$

So only A_{11}, A_{22}, and A_{33} are involved in the solution operations. The off-diagonal blocks are ONLY used in matrix-matrix multiplies.



Linear Programming; BP1600



BP 1600

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, reference 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**

PARALLELIZATION of MARKOWITZ/THRESHOLD CODE

Factorization times (secs) for code of Van der Stappen, Bisseling and Van de Vorst (1993) on a PARSYTEC SuperCluster FT-400

Matrix	Order	Entries	Number of processors 1	16	100	400
SHERMAN 2	1080	23094	1592	108	32.7	15.6
LNS 3937	3937	25407	2111	168	37.9	23.7

Factorization times for SPLU code of Bisseling and Koster (1997)

Matrix	p=1	p=4	p=9	p=16	p=25
PSMIGR 1		26.8	12.5	7.9	5.9
ONETONE1		57.1	27.7	17.5	10.6
ONETONE2	37.0	14.4	8.1	5.7	4.2
GOODWIN		66.0	31.9	29.4	20.4

Time (in secs) on p processors of a CRAY T3E

Matrix	p=2	p=4	p=6	p=8	p=10	p=12	p=14
PSMIGR 1		89.2	63.5	51.1	41.4	38.3	33.9
ONETONE1		125.2	84.6	77.7	58.2	49.4	51.3
ONETONE2	44.0	30.1	25.6	20.1	20.5	19.1	17.7
TWOTONE		1012.9	517.2	381.8	274.6	248.7	232.8
LHR71		351.0	260.5	249.6	209.9	177.1	144.8
GOODWIN	157.9	106.6	89.6	71.4	55.6	54.5	75.9
AV41092		1398.7	927.2	532.8	427.5	394.9	339.4

Time (in secs) on p processors of a MEIKO CS2-HA

TECHNIQUES FOR USING FULL BLAS IN SPARSE SOLUTION

There are three main approaches

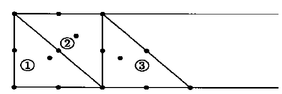
- **Frontal methods**
- **Multifrontal methods**
- **Supernodal methods**

FRONTAL METHODS

$$\mathbf{A} = \Sigma\, \mathbf{B}^{(l)}$$

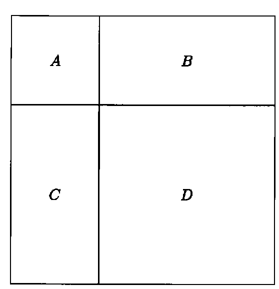
$$a_{ij} \leftarrow a_{ij} + b_{ij}^{(l)}$$

$$a_{ij} \leftarrow a_{ij} - a_{ik}[a_{kk}]^{-1}a_{kj}$$



Keep front in a full matrix

KERNEL OF FRONTAL CODES



$\mathbf{D} \leftarrow \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$

$\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ .... Full Matrices

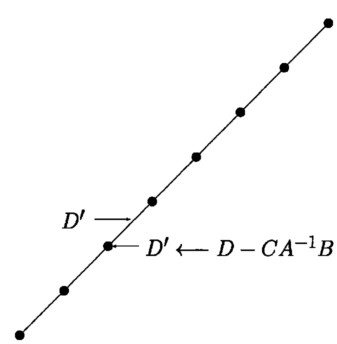
FRONTAL METHODS ON NON-ELEMENT PROBLEMS

x x   x
x x   x
0
.
.
.
.
0

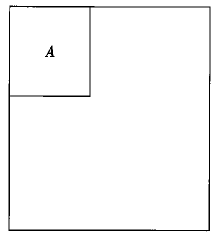
After the first two rows are 'summed', variable 1 does not appear in any subsequent equation. We can therefore eliminate variable 1 from the rectangular frontal matrix. Note that we can use partial pivoting in this case.

x x x 0
x x 0 x

REPRESENT FACTORIZATION
BY COMPUTATIONAL TREE



Pivoting within frontal matrix



A is fully assembled block from which pivots are chosen

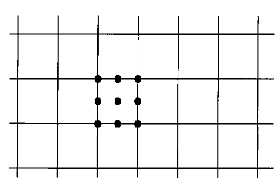
Threshold pivoting

$$|a_{lk}| \geq u.\max_i |a_{ik}|, \text{ where } u \in (0,1]$$

and

k is over columns of **A**
l is over rows of **A**
i is over whole frontal matrix

NINE-NODE ELEMENTS
'5' VARIABLES/NODE



PERFORMANCE OF FRONTAL CODE

	Grid size	Order	Mflop/s	GEMM Mf
CRAY 2 ..	100 x 100	202005	345	449
CRAY YMP ..	96 x 96	186245	272	313

Maximum order frontal matrix ~ 1000

Mflop/s includes i/o

LARGE PROBLEMS ON LARGE COMPUTERS

Matrix of order 202005 on CRAY-2
… 20 min (rate of 345 Mflop/s)

Largest problem of class …
20 million x 20 million

would require
1 month of 1 Gflop/s

and would produce
500 Gwords of data

EFFECT OF ORDERING

MC43 is routine for frontwidth reduction.
Problem: LOCK3491 with 684 elements.

Factorization performance

Without MC43: 122 Mflop/s
With MC43: 80 Mflop/s

But ….

Time (secs) are:
Without MC43: 9.36 secs
With MC43: 1.17 secs

Maximum frontsize reduces from 836 to 217

MAJOR LIMITATION

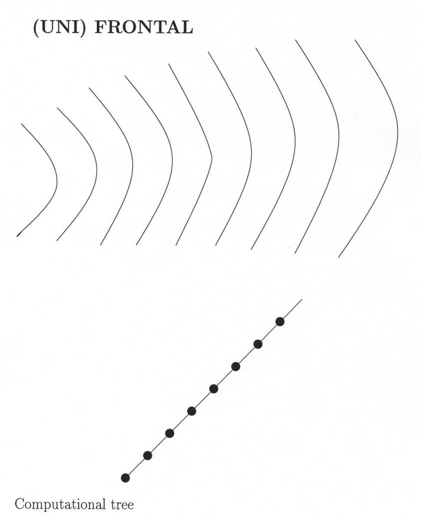
Mflop/s are high but flops are also!

Can extend applicability by reordering but still not suitable for general systems

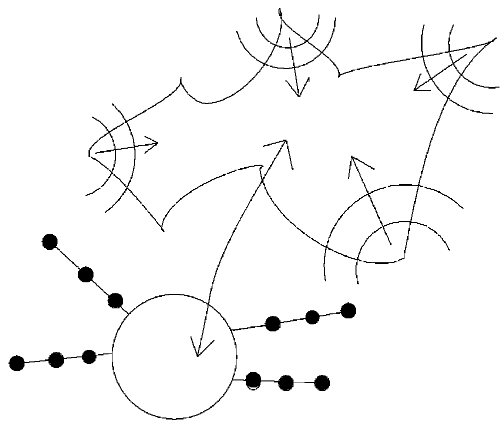
Second problem with frontal schemes relates to

PARALLELISM

or rather lack of it



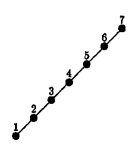
MULTIFRONTAL



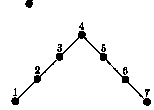
TRIDIAGONAL MATRIX

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

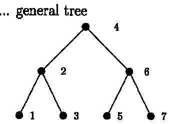
Frontal ... chain



BABE



Multifrontal ... general tree



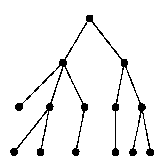
M
Ultifrontal
Parallel
Solver

Amestoy + Duff

CERFACS

Designed for *nearly* symmetric systems

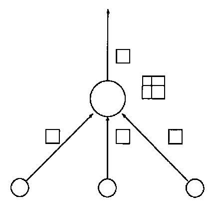
ASSEMBLY TREE



AT EACH NODE
Perform some steps of Gaussian Elimination of Full Matrices
(Frontal Matrices)

A	B
C	D

$D \leftarrow D - CA^{-1}B$



Assembly ...
Gather/scatter operations
Indirect addressing

Elimination ...
Full Gaussian elimination
Level 3 BLAS (TRSM, GEMM)

Perform the analysis on the pattern of $\mathbf{A} + \mathbf{A}^T$

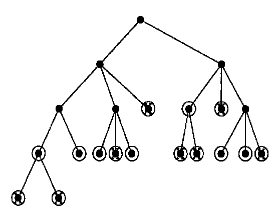
Can generate assembly tree symbolically using any sparsity preserving ordering

Then perform numerical pivoting *a posteriori* on the tree.

Performance in Mflop/s of multifrontal code MA41 on matrix BCSSTK15 on range of RISC processors

Computer	Peak	DGEMM	MA41
DEC 3000/400 AXP	133	49	34
HP 715/64	128	55	30
IBM RS6000/750	125	101	64
IBM SP2 (Thin node)	266	213	122
MEIKO CS2-HA	100	43	31

PARALLEL IMPLEMENTATION



Work corresponding to leaf nodes can proceed immediately and independently

When all children complete then work can commence at parent node.

STATISTICS ON FRONT SIZES IN TREE

Matrix	Order	Tree nodes	Leaf nodes No.	Leaf nodes Av. size	Top 3 levels No.	Top 3 levels Av. size
BCSSTK15	3948	576	317	13	10	376
BCSSTK33	8738	545	198	5	10	711
BBMAT	38744	5716	3621	23	10	1463
GRE1107	1107	344	250	7	12	129
SAYLR4	3564	1341	1010	5	12	123
GEMAT11	4929	1300	973	10	112	148

Typically 75% work in top three levels

TWO FORMS OF PARALLELISM

1. Tree parallelism [from assembly tree]

2. Node parallelism

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

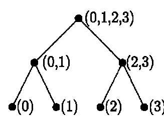
as in full linear algebra (eg, LAPACK)

SHARED MEMORY PARALLEL PERFORMANCE

MA41 on 7 processors of a CRAY C98
In columns (1) we exploit only parallelism from the tree
In columns (2) we combine the two levels of parallelism

Matrix	order	entries	(1) Mflop/s	(1) (speedup)	(2) Mflop/s	(2) (speedup)
WANG3	26064	177168	1062	(1.42)	3718	(4.98)
WANG4	26068	177196	1262	(1.70)	3994	(5.39)
BBMAT	38744	1771722	2182	(3.15)	3777	(5.46)

DISTRIBUTED MEMORY



PARASOL

PARASOL - Integrated Environment for Parallel Sparse Matrix Solvers

URL: http://www.genias.de/parasol.html

Partners

- **European research organizations: CERFACS, GMD-SCAI, ONERA, RAL, Para//abb**
- **Industrial code developers: INPRO, MacNeal-Schwendler, Det Norske Veritas, Polyflow, Apex Technologies**
- **European HPC software companies: GENIAS, PALLAS**

OBJECTIVES

- **Develop parallel sparse matrix solvers.**
- **Specify an open library interface (PARASOL Library).**
- **Implement portable prototypes on different HPC systems.**
- **Integrate the solvers and additional tools into TRAPPER.**
- **Evaluate the PARASOL library and utilities.**

Final Library will be public domain

Main target machines are:

- **IBM SP-2**
- **SGI Origin 2000**

with tests also on:

- **CRAY T3D/E**
- **Fujitsu VPP**

We are coding in Fortran 90 with MPI

The direct solver in PARASOL is MUMPS:

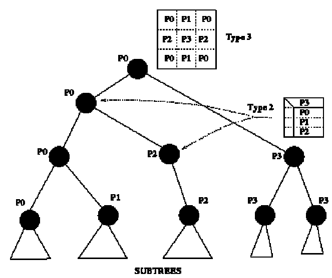
M
Ultifrontal
Massively
Parallel
Solver

Patrick R Amestoy (ENSEEIHT)
Iain S Duff (CERFACS/RAL)
Jean-Yves L'Excellent (CERFACS/ENSEEIHT)
Jacko Koster (RAL)
Petr Plecháč (RAL/Delaware)
Miroslav Tůma (CERFACS/Prague)

mumps@cerfacs.fr

Distributed memory parallelism

- Type 1: Parallelism of the tree
- Type 2: 1D partitioning of frontal matrices with distributed assembly process
- Type 3: 2D partitioning of root modes (ScaLAPACK)



Statistics for two PARASOL test problems

Matrix	Order	Nonzeros in matrix ($x10^6$)	Nonzeros in factors ($x10^6$)	Flops ($x10^9$)	Size of root	Time for analysis (on SP2)
SYMMETRIC CODE						
QUER	59122	1.5	12	4.0	1043	3.0
CRANKSEG2	63838	7.1	73	101.9	3127	14.5
UNSYMMETRIC CODE						
QUER	59122	3.0	20	8.0	1043	5.5

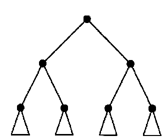
Performance of the unsymmetric and symmetric versions of the code on test problem QUER on an IBM SP2

Working processors	Time for factorization	
	unsymmetric	symmetric
1 (CPU)	64.9	41.1
1 (elapsed)	299.2	150.4
2 (elapsed)	109.1	21.0
4 (elapsed)	19.1	12.9
8 (elapsed)	15.2	9.3
16 (elapsed)	11.5	6.4
24 (elapsed)	10.1	6.6
32 (elapsed)	10.5	5.8

Results for the symmetric version of the code on CRANKSEG2

Machine	Working processors	Time for factorization
SP2	16	42
	24	31
	32	27
Origin	1	217
	2	112
	4	66
	8	46
	16	29

Incomplete Nested Dissection



Order subtrees using minimum degree

Effect of ordering on Problem BMWCRA-1
Order 148,770. Entries 5,396,386

Analysis

	Entries in factors	Operations
AMD	113,788,395	1.28×10^{11}
ND/MMD	85,301,445	6.62×10^{10}

Factorization [Origin 2000 … Bergen]
Time in seconds (speedups in parentheses)

	Factor		Solve	
No. procs	AMD	ND+MMD	AMD	ND+MMD
1	687	307	12.0	10.1
2	408	178	7.5	5.4
4	236	82	6.7	4.2
8	143(4.8)	58(5.3)	4.2(2.9)	2.6(3.9)
16	112	36	2.9	1.9

Time (in seconds) for factorization using MUMPS 4.0 with default options on IBM SP2. ND ordering is used.
(*): uniprocessor CPU or estimated CPU time.
-: not enough memory.

Matrix	Number of processors					
	1(*)	4	8	16	24	32
OILPAN	33	11.1	7.5	5.2	4.8	4.6
B5TUER	108	82.1	51.9	13.4	13.1	10.5
CRANKSEG_1	270	185.3	92.4	27.3	25.6	20.9
CRANKSEG_2	378	-	-	41.8	31.0	27.2
BMW7ST_1	104	-	29.8	13.7	11.7	11.3
BMW3_2	246	-	-	24.1	24.0	20.4
MIXING-TANK	104	30.8	21.6	16.4	14.7	14.8
INV-EXTRUSION-1	70	24.9	16.3	13.5	13.4	12.4
BBMAT	198	255.4	85.2	34.8	32.8	30.9

Time (in seconds) for factorization using MUMPS 4.0 with default options on SGI Origin 2000. ND or MFR ordering is used.

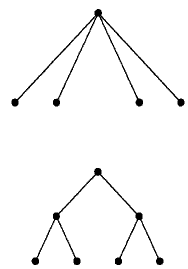
Matrix	Number of processors				
	1	2	4	8	16
CRANKSEG_2	217	112	66	46	29
BMW7ST_1	62	36	25	12	10
BMWCRA_1	307	178	82	58	36
BMW3_2	151	96	53	33	18
M_T1.RSE	92	56	31	19	13
SHIP_003.RSE	392	237	124	108	51
SHIPSEC1.RSE	174	125	63	39	25
SHIPSEC5.RSE	281	181	103	62	37
SHIPSEC8.RSE	187	119	64	35	27
THREAD.RSE	186	125	70	38	24
X104.RSE	56	34	19	12	11

Multiple Fronts

Domain Decomposition

One level

Multilevel



MULTIPLE FRONTS

Work …. Regular 2N x 2N grid

Single region: $32N^4 + O(N^3)$

Four regions: $18.6N^4 + O(N^3)$

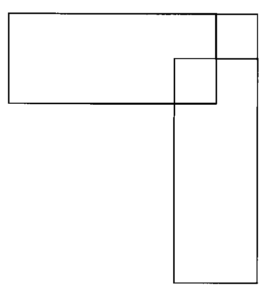
Number subdomains	Flops (10^6)	CPU time (secs)
1	16790	69.4
4	10350	48.4
8	8365	40.3

Results on one-processor of CRAY Y-MP8I [48 x 48 grid]

Number Subdomains	CRAY Y-MP Wall clock (secs)	CRAY Y-MP Speedup	DEC Alpha (w/PVM) Wall clock (secs)	DEC Alpha (w/PVM) Speedup
1	98.75	-	1460.3	-
2	64.57	1.5	1043.0	1.4
4	30.65	3.2	457.5	3.2
8	15.25	6.5	-	-

Multiprocessor multiple front runs [48 x 48 grid]

Unsymmetric Multifrontal



UNSYMMETRIC MULTIFRONTAL

- **Assembly tree no longer suitable model for computation. Use directed acyclic graph *dag*.**
- **Frontal matrices no longer square.**
  Use rectangular frontal matrices.
- **Fronts persist in dag rather than being absorbed by parent node.**

  More complicated data structures.
- **Very sensitive to numerical pivoting.**
  Pay attentions to numerical values when doing analysis

Code
UMFPACK/MA38 …. Davis and Duff

See Report:

A combined unifrontal/multifrontal method for unsymmetric sparse matries.
Timothy A Davis and Iain S Duff
Report RAL-TR-97-046, Rutherford Appleton Laboratory
http://www.numerical.rl.ac.uk/reports/reports

Comparison of "symmetric" (MA41) and "unsymmetric" (MA38) code

Order	13535	62424	26064	22560	120750	36057
Nonzeros	390607	1717792	177168	1014951	1224224	227628
Index of asymmetry	0.00	0.00	0.00	0.36	0.76	0.89
Floating-point ops (10^9)						
MA41	0.3	2.3	10.4	2.9	38.2	0.6
MA38	3.8	5.3	62.2	9.0	7.0	0.2
Factorization time						
(*seconds on Sun ULTRA*)						
MA41	8	46	174	81	809	19
MA38	85	127	1255	226	220	11

$$\text{Index of asymmetry} = \frac{\text{Number of pairs such that } a_{ij}=0, a_{ji}\neq 0}{\text{Total number of off-diagonal entries}}$$

Comparison of MA38 with MA48 and MA41 on a few test matrices.

Times in seconds on a Sun ULTRA-1 workstation.

Matrix	Order	No entries	Factorization Time		
			MA38	MA48	MA41
AV41092	41092	1683902	1618	1296	254
PSMIGR_1	3140	543162	198	179	188
ONETONE1	36057	341088	57	110	189
LHR71C	70304	1528092	93	287	996

Comparison of multifrontal against supernodal approach

Times in seconds on a Sun ULTRA-1 workstation

Matrix	Order	Entries	Analyse & Factorize Time (secs)		Entries in LU factors (10^6)	
			SuperLU	Multif	SuperLU	Multif
ONETONE2	36057	227628	9	11	1.3	1.3
TWOTONE	120750	1224224	758	221	24.7	9.8
WANG3	26024	177168	1512	174	27.0	11.4
VENKAT50	62424	1717792	172	46	18.0	11.9
RIM	22560	1014951	78	80	9.7	7.4
GARON2	13535	390697	60	8	5.1	2.4

COMPARISON OF SPARSE DIRECT APPROACHES

- **Frontal codes**
  - **Require a well ordered matrix**
  - **Good on vector machines**
  - **Good for one-off solution**
  - **Can easily be run with little main memory**
- **Multifrontal codes**
  - **Good for reducing number of entries in factors**
  - **Efficient if significant fill-in**
  - **Natural way to parallelize method**
- **Markowitz/threshold**
  - **Good for very sparse systems**
  - **Good when structure very irregular**
  - **Good at keeping storage requirements low**
  - **Often good for repeated solutions**
  - **Not easy to parallelize**

SUMMARY

- **There is no one best buy**
- **The best code in any situation will depend on**
  - **The solution environment**
  - **The computing platform**
  - **The structure of the matrix**

In short, it is

HORSES FOR COURSES