Science & Technology Facilities Council

The use of direct methods in the solution of sparse linear equations for constrained optimization

Iain Duff

STFC Rutherford Appleton Laboratory and CERFACS

ICIAM 2011.
Vancouver Conference Centre, Canada.
July 17-24 2011

Direct methods

Gaussian Elimination

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

Permutations **P** and **Q** chosen to preserve sparsity and maintain stability

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

SOLVE:

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

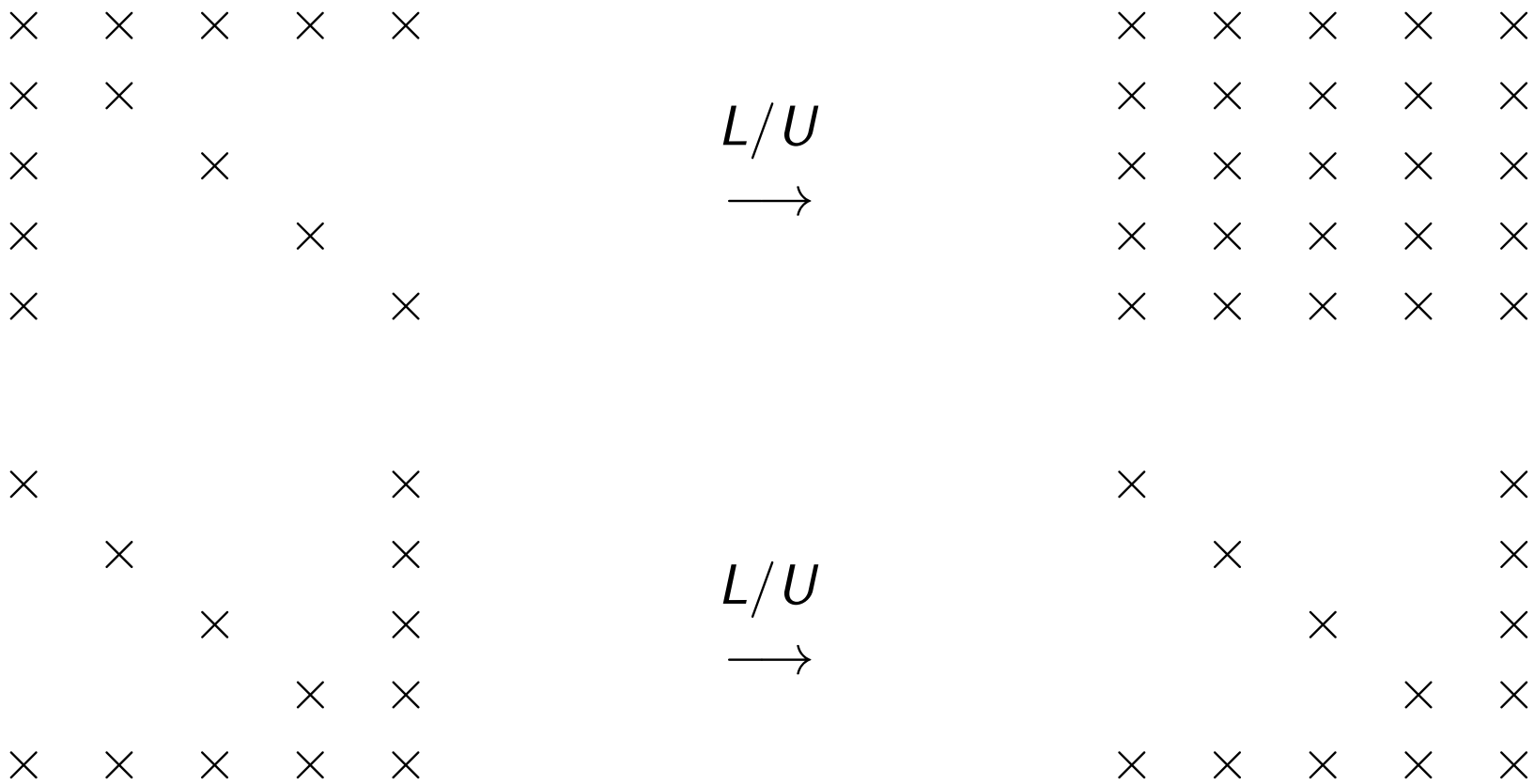
by

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

then

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

Direct methods ... Ordering for Sparsity



MINIMUM DEGREE (Tinney S2 ... 1967)

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

Direct methods

A sparse direct method normally consists of three phases

- Analysis (determine ordering and data structures)
- Numerical factorization ($A \longrightarrow LDL^T$)
- Solution phase (obtain solution using sparse triangular solves)

Direct methods

When the matrix is positive definite this works well but in the indefinite case subsequent numerical pivoting may mean that the initial analysis is not respected.

The very worst example of a difficult indefinite system is an augmented system that, for example, arises in the case of constrained optimization problems. The matrix will typically have the form:

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix}$$

where H is (an approximation to) the Hessian and A is the matrix of constraints.

Pivoting for augmented systems

The indefiniteness is handled well by block two-by-two pivoting [Bunch, Kaufman, Parlett]. However, the structure is the problem.

There have been a great number of attempts to tame this problem by choosing pivots that exploit the structure of these systems ... or at least respect it in some sense.

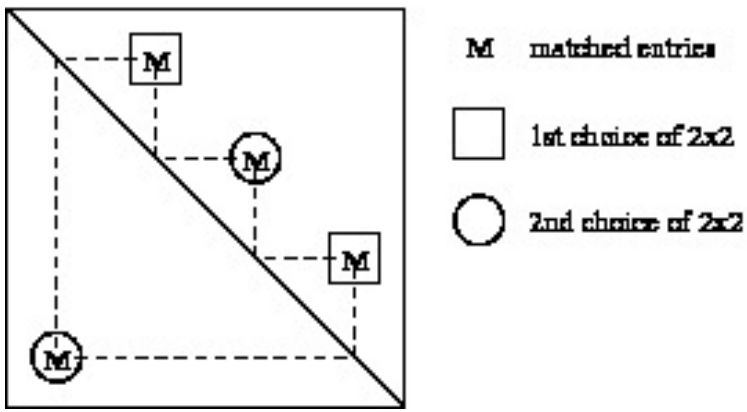
Within HSL alone, we have tried: `MA47`, `MA67`, and ``‘‘MA49’’``(Gould).

Many others have also tried.

One idea that I like is to use compressed graphs.

Pivoting for augmented systems

First identify a strong transversal (set of entries with no two in same row or column), then associate these as a potential two-by-two pivot viz.



where the transversal entries were identified by using MC64 that finds a transversal to maximize the product of the absolute value of the transversal entries.

Pivoting for augmented systems

Does it work?

Pivoting for augmented systems

Does it work?

Results from runs by Stéphane Pralet.

Entries in factors ($\times 10^4$)

Matrix	MeTiS		CMP
	Forecast	Actual	
BRATU3D	556.9	1148.4	562.1
crystk02	443.2	443.2	432.0
stokes128	275.4	343.7	453.2
DTOC	18.8	471.4	1291.8

Pivoting for augmented systems

Does it work?

Results from runs by Stéphane Pralet.

Entries in factors ($\times 10^4$)

Matrix	MeTiS		CMP
	Forecast	Actual	
BRATU3D	556.9	1148.4	562.1
crystk02	443.2	443.2	432.0
stokes128	275.4	343.7	453.2
DTOC	18.8	471.4	1291.8

It can work ... but can also be very bad

Static Pivoting

The default action for general matrices is to use some form of threshold pivoting in the numerical factorization phase. This normally increases both storage and work from that forecast by the analysis.

An alternative is to use **Static Pivoting**, by replacing potentially small pivots p_k by

$$p_k + \tau$$

and maintaining the same pivoting strategy as advocated in the analysis.

This is even more important in the case of parallel implementation where static data structures are often preferred

Static Pivoting

Several codes use (or have an option for) this device:

- SuperLU (Demmel and Li)
- PARDISO (Gärtner and Schenk)
- MA57 (Duff and Pralet)
- MUMPS (Amestoy, Duff, L'Excellent, and Koster)

mumps.enseeiht.fr mumps@cerfacs.fr

Static pivoting

Matrix DTOC of order 24993 with 34986 entries

Runs using HSL package MA57

Factorization time seconds		Entries in the factors ($\times 10^6$)	
threshold	static	threshold	static
29.1	0.41	4.71	.19

Static Pivoting

We have factorized

$$A + E = LDL^T$$

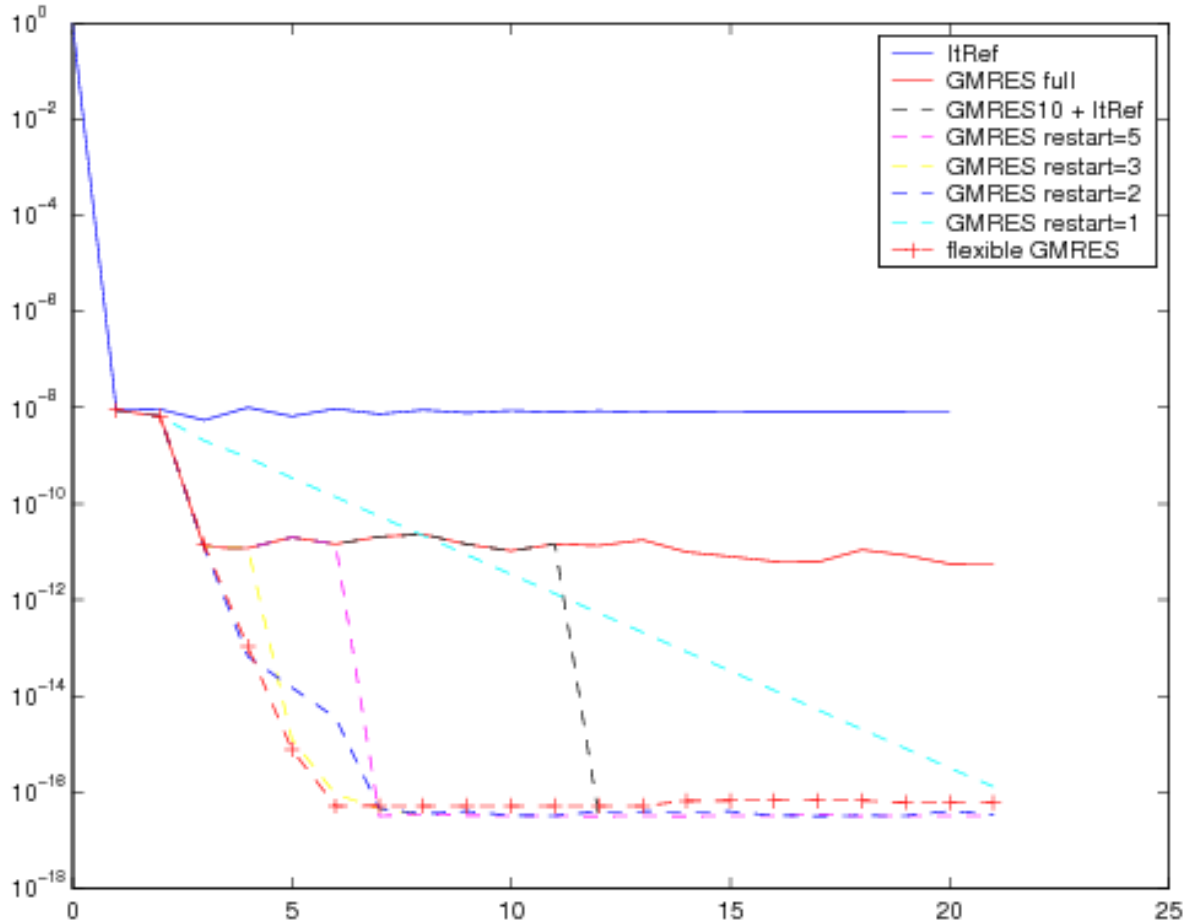
where $|E| \leq \tau I$

The four codes then have an **Iterative Refinement** option

The problem is that this sometimes does not converge
also
Static pivoting does not respect inertia

Arioli, Duff and Gratton have shown that using FGMRES rather than iterative refinement results in a backward stable method that converges for really quite poor factorizations of A.

Numerical experiments



Restarted GMRES vs. FGMRES on CONT-201 test example:
$\tau = 10^{-8}$

Constraint preconditioner

Use of direct methods in constraint preconditioner
[Dollar, Gould, Schilders, and Wathen (2006)]

When solving the augmented system with coefficient matrix

$$\begin{pmatrix} \mathbf{A} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix}$$

a possible constraint preconditioning matrix might be of the form

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{B}_1^T \\ \mathbf{0} & \mathbf{G} & \mathbf{B}_2^T \\ \mathbf{B}_1 & \mathbf{B}_2 & \mathbf{0} \end{pmatrix}$$

where $\mathbf{B}_1$ is nonsingular.
The solution of the preconditioned system then involves solving for $\mathbf{G}$ and $\mathbf{B}_1$ (and $\mathbf{B}_1^T$).

Constraint preconditioner

Normally (and in the HSL implementation HSL_MI13), a direct method is used both to obtain the nonsingular $\mathbf{B}_1$ from $\mathbf{B}$ and to solve for $\mathbf{G}$ and $\mathbf{B}_1$ (and $\mathbf{B}_1^T$). In HSL_MI13, HSL_MA48 is used for the rectangular and unsymmetric matrices and MA57 for the symmetric matrix $\mathbf{G}$.

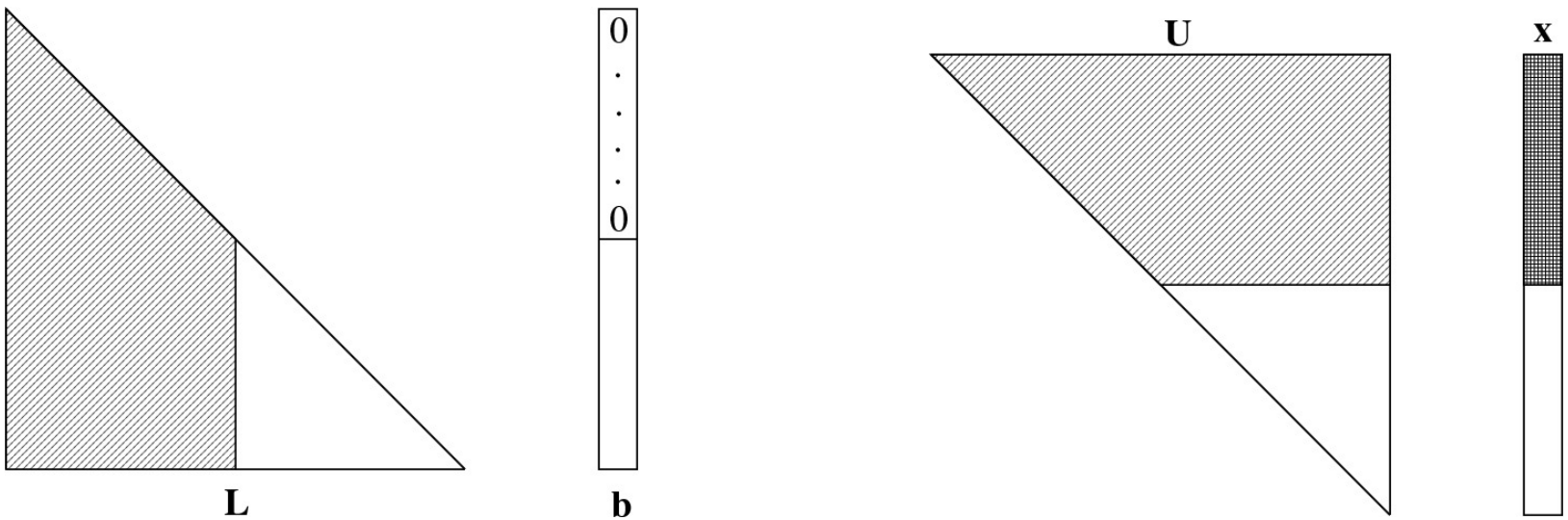
Constraint preconditioner

Results of some runs on CUTEr QP problems
[Dollar, Gould, Wathen]

	MA57			Implicit factors		
Name	Fact time	Its	Total time	Fact time	Its	Total time
CONT5-QP	3.37	1	3.83	19.94	37	22.20
AUG2DCQP	0.46	1	0.53	0.25	125	2.01
CVXQP3	9.93	0	10.13	0.34	43	0.68
DEGENQP	14.36	1	14.72	2.45	3	2.89

Solving least-squares problems

An observation in Chapter 7 of the book by Duff, Erisman, and Reid (1986) notes that savings can be made if there are zeros in the right-hand side and if only part of the solution is required. Specifically



where the shaded area of the factors need not be kept or used in the solution.

Solving least-squares problems

The augmented system representation for the least-squares problem can be written as:

$$\begin{pmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}$$

so that by choosing the early pivots from the last m rows of the matrix and the last pivots from the last m columns, we can exploit both of these savings.

Solving least-squares problems

Results from Duff and Reid (1976).

Number of multiplications in `SOLVE` phase for some least-squares problems.

Number of rows	200	219	331
Number of columns	199	85	104
Number of nonzeros	702	438	662
Using symmetry	3748	2046	3103
Ignoring symmetry	2581	1441	2109

This shows that on these very small problems significant savings can be made by ignoring symmetry in work and storage.

Solving least-squares problems

We can use HSL_MA48 to develop an algorithm that exploits this observation.

The following description was developed after a coffee time discussion last week with Arioli, Gould, and Thorne at RAL.

The LU factorization of $\mathbf{A}$ by HSL_MA48 can define the partition $\mathbf{A}^T = \mathbf{A}_1^T \mathbf{A}_2^T$ where $\mathbf{A}_1$ is square and nonsingular.
We can then write the least-squares augmented system as:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_1 \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_2 \\ \mathbf{A}_1^T & \mathbf{A}_2^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{0} \end{pmatrix}$$

Solving least-squares problems

As we are using an unsymmetric solver, it is better to show the blocking as:

$$\begin{pmatrix} \mathbf{A}_1^T & \mathbf{A}_2^T & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{A}_1 \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

We then see clearly that because of the zero block in the right-hand side the factorization of the first block of columns is not needed and, because we need not solve for $\mathbf{r}_1$, the first block rows are not needed either.

Solving least-squares problems

We thus need only concern ourselves with the solution of the Schur complement system:

$$\begin{pmatrix} \mathbf{A}_1^{-T}\mathbf{A}_2^T & \mathbf{A}_1 \\ \mathbf{I} & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

and if we pivot first on the first $(m-n)$ columns then we can avoid storing or using the first n rows of the LU factorization of this matrix as we do not need to compute $\mathbf{r}_2$.

Solving least-squares problems

As we already computed the LU factorization of $\mathbf{A}$, we can express the reduced system in the form:

$$\begin{pmatrix} \mathbf{L}_1^{-T}\mathbf{L}_2^T & \mathbf{L}_1 \\ \mathbf{I} & \mathbf{L}_2 \end{pmatrix} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

where $\mathbf{y} = \mathbf{U}\mathbf{x}$ and $\mathbf{A}_1 = \mathbf{L}_1\mathbf{U}$ and $\mathbf{L}_2$ is the rectangular part of the L-factor of $\mathbf{A}$.

Solving least-squares problems

The system can also be written as:

$$\begin{pmatrix} \mathbf{L}_1^{-T} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{L}_2^T & \mathbf{L}_1^T \mathbf{L}_1 \\ \mathbf{I} & \mathbf{L}_2 \end{pmatrix} \begin{pmatrix} \mathbf{r}_2 \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix}$$

Solving least-squares problems

I had a frustrating and exhausting time on Friday trying to write and debug a program to do this using HSL_MA48 and had some mixed results.

Solving least-squares problems

I had a frustrating and exhausting time on Friday trying to write and debug a program to do this using `HSL_MA48` and had some mixed results.

However, with the matrix **A** as matrix `mesh_deform.rb` from Tim Davis' Florida collection [234023 $\times$ 9303 with 853829 entries], the number of entries in the relevant factors were:

Solving least-squares problems

I had a frustrating and exhausting time on Friday trying to write and debug a program to do this using HSL_MA48 and had some mixed results.

However, with the matrix **A** as matrix mesh_deform.rb from Tim Davis' Florida collection [234023 $\times$ 9303 with 853829 entries], the number of entries in the relevant factors were:

Exploiting symmetry	
Using MA57	11293712
Ignoring symmetry	
Using HSL_MA48	5026683

Conclusions

We have come a long way **but** ...

Conclusions

We have come a long way **but ...**

Many challenges and much research still to be done in this area

THANK YOU FOR YOUR ATTENTION

SPARSE DAYS at CERFACS

September 6th and 7th, 2011

Registration (free) and low cost accommodation by: 22 August

Web site: http://www.cerfacs.fr/6-26303-Upcoming-Meetings.php