Improving the performance of direct solvers for sparse symmetric indefinite linear systems

Jonathan Hogg and Jennifer Scott
STFC Rutherford Appleton Laboratory

IMA NLAO Birmingham, 11th September 2012

Sparse indefinite system

Solve

$$Ax = b$$

with A large, sparse, symmetric and indefinite.

For example, systems arise in a number of important applications

$$\begin{pmatrix} H & B^T \\ B & \delta I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}$$

(see next talk).

Direct method

- Compute explicit factorization

$$A = LDL^T$$

where L (unit) is lower triangular, D (block) diagonal.

- Complete solution by performing triangular solves.

Test examples

In this talk, we focus on tough indefinite systems only.

Examples from University of Florida Sparse Matrix Collection.

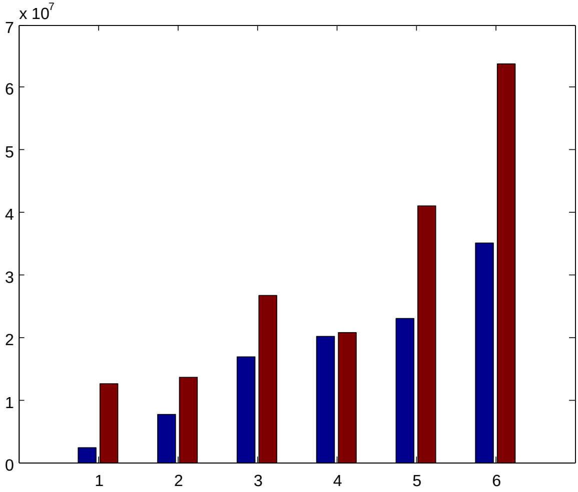
Identifier	n	$nz(A)$	$nz(L)$	*flops*
1. GHS_indef/ncvxqp1	12 111	73 963	1.68×10^6	7.28×10^8
2. GHS_indef/bratu3d	27 792	173 796	6.28×10^6	4.42×10^9
3. GHS_indef/cont-300	180 895	988 195	1.17×10^7	2.96×10^9
4. GHS_indef/d_pretok	182 730	1 641 672	1.46×10^7	5.06×10^9
5. TSOPF/TSOPF_FS_b300_c2	56 814	8 767 466	2.14×10^7	8.96×10^9
6. TSOPF/TSOPF_FS_b300_c3	84 414	13 135 930	3.31×10^7	1.43×10^{10}

* $nz(L)$ and *flops* are for positive definite equivalent with nested dissection ordering

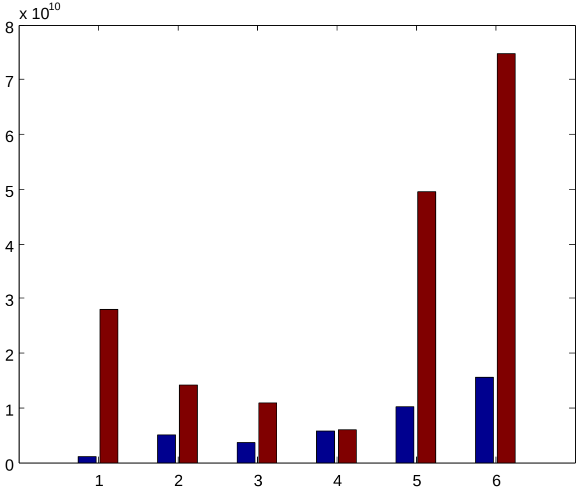
Let's look at the problem ...

- Run indefinite solver.
- Put large entries on diagonal and run positive definite solver.
- Compare the performance.

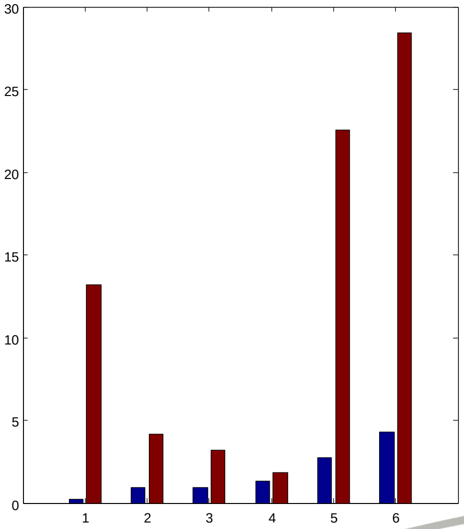
Positive definite versus indefinite $nz(L)$



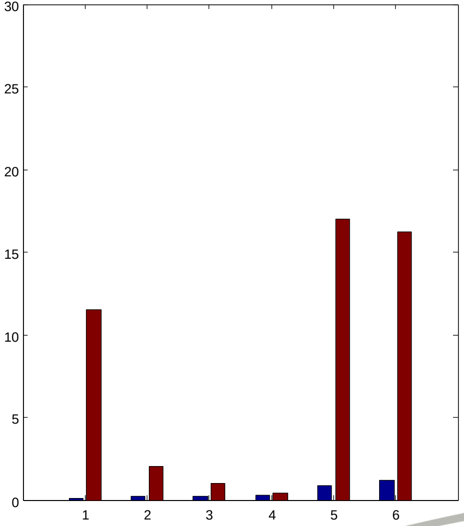
Positive definite versus indefinite flops



Positive definite versus indefinite time (serial)



Positive definite versus indefinite time (8 cores)



Why the differences?

- May not be able to use pivot sequence in supplied order.
- Rejected pivots $\Rightarrow$ more flops and denser factors.
- Extra data movement.
- Less scope for parallelism.

Our aim: improve indefinite performance without compromising stability or the computation of the inertia.

Note: indefinite solver designed on assumption of few rejected pivots so we need to reduce rejected pivots.

The heart of a direct solver

Elimination (pivot) order pre-selected to reduce fill in.

At each stage of factorization, the solver works with dense $m \times m$ submatrix ($m \ll n$)

$$\begin{pmatrix} F_1 & F_2^T \\ F_2 & E \end{pmatrix}.$$

Only rows/columns of F_1 are ready for elimination.

- Factorization: $F_1 = L_1 D L_1^T$
- Solve: $L_2 = F_2 L_1^{-1}$. (L_1, L_2) are computed columns of L.
- Update: $E \leftarrow E - L_2 (L_2 D)^T$ (BLAS 3).

Achieving good solver performance

Key is efficiency of partial dense factorizations.

In positive-definite case:

- Pivots can be selected from diagonal of F_1 in turn ... allows data structures to be fixed before factorization commences (simplifies code and reduces data movement).
- Factorization of F_1 can begin before all updates to F_2 have been made (improves scope for parallelism ... work with block tasks).

Indefinite case

For good performance want to use the supplied pivot sequence.

But

- Zero (or small) diagonal entries cannot be used as pivots.
- Necessary to incorporate numerical pivoting.
- 1×1 and 2×2 pivots needed to retain symmetry.
- Standard approach: threshold partial pivoting.

Threshold partial pivoting

Involves checking that the candidate pivot is 'large' compared to the other entries in its column(s).

Test for 1×1 pivot:

$$|a_{q+1,q+1}| > u \max_{q+1<i\leq n} |a_{i,q+1}|.$$

Corresponding test for 2×2 pivot:

$$\left| \begin{pmatrix} a_{q+1,q+1} & a_{q+1,q+2} \\ a_{q+1,q+2} & a_{q+2,q+2} \end{pmatrix}^{-1} \right| \begin{pmatrix} \max_{q+2<i\leq n} |a_{i,q+1}| \\ \max_{q+2<i\leq n} |a_{i,q+2}| \end{pmatrix} < \begin{pmatrix} u^{-1} \\ u^{-1} \end{pmatrix}.$$

Threshold partial pivoting

- u is threshold parameter, typical default value 0.01. This was used in our earlier tests.
- Larger u favours stability; smaller u means fewer rejects.
- If a pivot fails test, may have to be delayed until later in factorization. This is what we want to avoid.

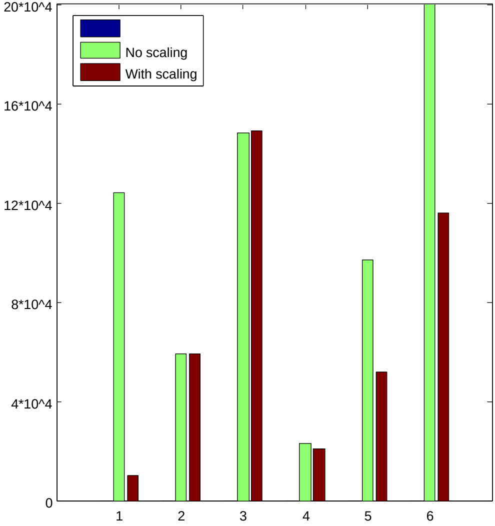
How can we reduce delays?

First remedy: scaling

- In particular, use symmetrized version of `MC64` (Duff and Koster, Duff and Pralet), which is based on maximum weighted matchings.

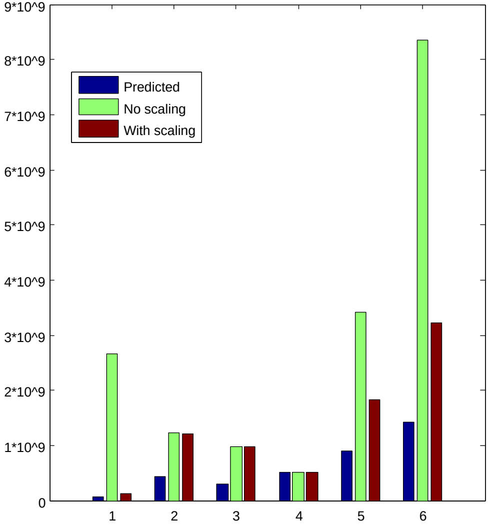
- Entries in scaled matrix SAS that are in the matching have absolute value 1 while rest have absolute value ≤ 1.

Effect of scaling on delayed pivots



Science & Technology Facilities Council

Effect of scaling on flops

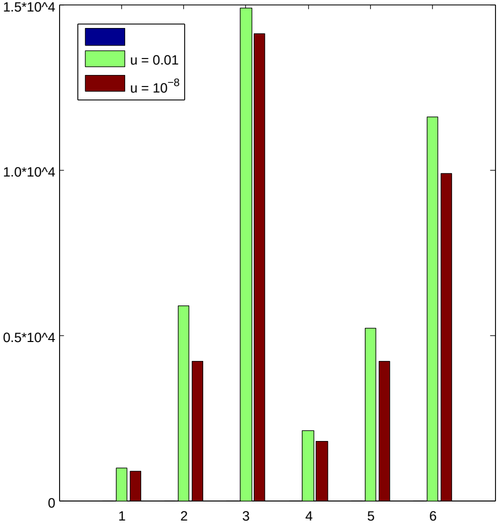


Next remedy: small u

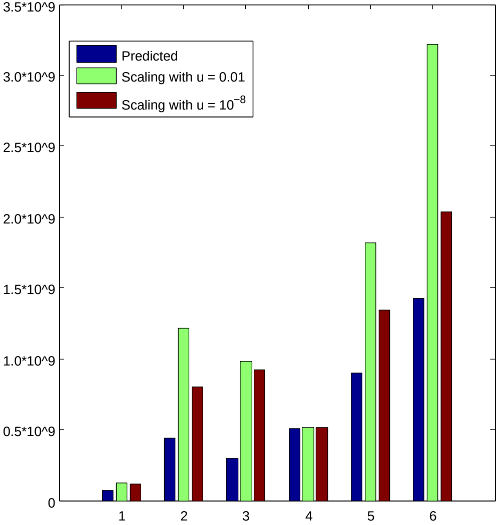
- Use a smaller threshold u to weaken stability test.
- If necessary, use iterative refinement or FGMRES to recover accuracy.
- If u too small, entries of L can become unbounded.
- Here we use $u = 10^{-8}$.

Note: in this and all other experiments, we prescale.

Effect of small u on delayed pivots



Effect of small u on flops



Story so far:

- Good scaling can really help.
- Small u may also help ... but may need additional solves.

So what else?

Try preselecting 2×2 pivots?

An approach that does this is MA47 (Duff and Reid):
sparse indefinite solver that uses structured 2×2 pivots.

Experiments show can work really well for matrices of form

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

but more generally leads to much denser factors
(without eliminating delayed pivots).

What else? Constraint ordering

Proposed (Bridson) for systems of form

$$\begin{pmatrix} H & B^T \\ B & C \end{pmatrix}$$

with H symmetric positive definite, B rectangular, and C symmetric positive semi-definite.

Only order a C-node after its H-node neighbours have been ordered.

Advantages: able to use modified Cholesky code with no delays (although stability not guaranteed, works in practice).

But: too restrictive so that generally much denser factors and more flops (can require order of magnitude more flops).

So what else? Matching orderings

Aim: permute large off-diagonal entries a_{ij} close to diagonal so that 2×2 block

$$\begin{pmatrix} a_{ii} & a_{ij} \\ a_{ij} & a_{jj} \end{pmatrix}$$

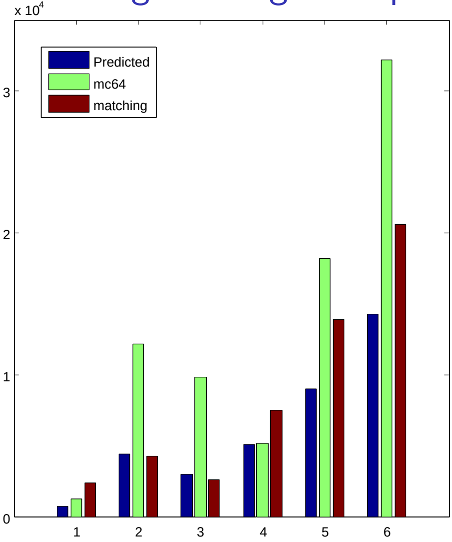
is potentially good 2×2 candidate pivot.

Use cycle structure of permutation associated with unsymmetric maximum weighted matching $\mathcal{M}$ to obtain such a permutation

(Duff and Gilbert, also Duff and Pralet, Schenk *et. al.*).

Combines scaling with ordering in single step.

Effect of matching ordering on flops



Science & Technology Facilities Council

Effect of matching ordering

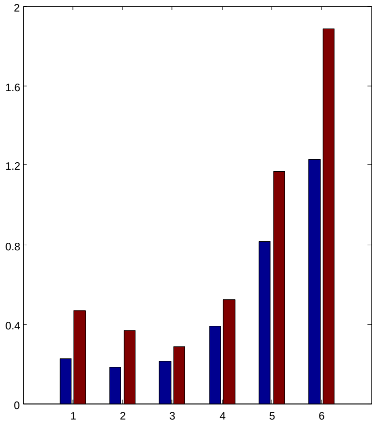
- Predicted values in last plot were for default ordering.
- Predicted values for matching ordering are typically 50 to 100% greater.

  But for the matching ordering, (almost) no delays and, most importantly,

  $$\text{predicted flops (and } nz(L)) \approx \text{actual flops (and } nz(L))$$

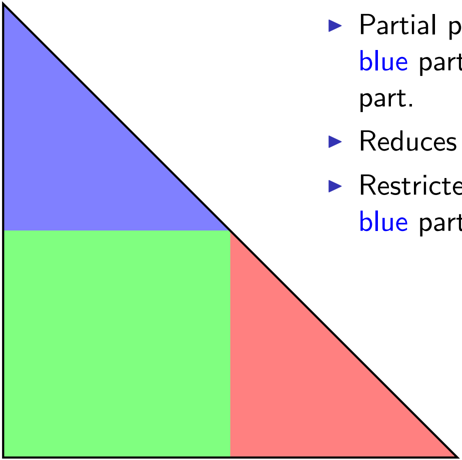
- Also, matching ordering stable (single step of refinement sufficient with $u = 0.01$ and 10^{-8}).

Positive definite versus indefinite time (matching ordering)



Difference now is down to pivot searches that restrict parallelism.

Restricted pivoting



- Partial pivoting — when factorizing blue part, takes into account green part.
- Reduces scope for parallelism.
- Restricted pivoting: just pivot within blue part.

Restricted pivoting

- Found that used just with scaling can lead to numerical instability (accuracy not recovered by refinement).
- If combined with matching ordering, works well for many problems
- But does not give stable factorization in all cases so not recommended for black box solver (note: it is used within PARDISO).

Concluding remarks

- Strategies explored to reduce delayed pivots and hence improve performance of direct solvers for tough (non-singular) indefinite problems.
- Robust approach: matching ordering (used with scaling), combined with threshold partial pivoting.
- But matching is expensive so only use on tough problems.
- Still requires access to whole pivot column and so scope for parallelism less than in positive-definite case.
- For many problems can get away with cheaper strategies but for a robust solver, matching is a good fall back strategy.

More details, further suggestions and lots of results available in technical report RAL-TR-2012-009.

Thank you!

Work supported ESPRC grant EP/I013067/1

Science & Technology Facilities Council