Science & Technology Facilities Council

Memory-Efficient Incomplete Factorizations for Sparse Symmetric Systems

Jennifer Scott

STFC Rutherford Appleton Laboratory

Miroslav Tůma

Institute of Computer Science
Academy of Sciences of the Czech Republic

Householder Symposium XIX, Spa, Belgium
June 8–13, 2014

Introduction

Focus in this talk is on saddle-point systems $Kx = b$ where

$$K = \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}$$

- A is $n \times n$ symmetric positive definite
- B is rectangular and of full rank
- C is $m \times m$ ($m \leq n$) symmetric positive semi-definite

Introduction

- Lots of papers on this in recent years ... too many names to mention.
- There are a number of direct solvers that are primarily designed for symmetric indefinite problems (including `MA57`, `HSL_MA86` and `HSL_MA97`) that perform well on these systems.
- For very large problems, need an iterative method and a good preconditioner. Many possibilities ... here we consider a class of incomplete factorization preconditioners.
- Recent related work includes Orban (2013) and Greif et al (see later).

Introduction

Consider the factorization

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -I \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix},$$

$$A = L_{11} L_{11}^T$$

$$L_{11} L_{21} = B$$

$$S = C + L_{21} L_{21}^T = L_{22} L_{22}^T$$

This factorization always exists without pivoting although numerical stability **not** guaranteed.

Introduction

If we set

$$\mathcal{L} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix},$$

then

$$P = \mathcal{L}\mathcal{L}^T$$

can be used as a (split) preconditioner.

$\mathcal{L}^{-1}K\mathcal{L}^{-T}$ has two distinct eigenvalues ± 1, so symmetric Krylov subspace method (MINRES or SYMMLQ) converges in at most two iterations.

Introduction

In practice, to obtain a preconditioner

$$A \approx \tilde{L}_{11}\tilde{L}_{11}^T \text{ (incomplete Cholesky)}$$

$$\tilde{L}_{11}^T\tilde{L}_{21} = B^T \text{ (triangular solve)}$$

$$\tilde{S} = C + \tilde{L}_{21}\tilde{L}_{21}^T \approx \tilde{L}_{22}\tilde{L}_{22}^T \text{ (incomplete Cholesky)}$$

Key question: how to sparsify $\tilde{L}_{21}^T$? (need $\tilde{S}$ sparse)

We will take a different approach that does **not** limit working to the $(1,1)$ block and then the $(2,2)$ block.

Constrained orderings

Consider for now the complete factorization of K.

Constrained ordering: find permutation Q such that QKQ^T can be factorized stably **without** numerical pivoting and **without** modifying entries in K, while maintaining sparsity.

Bridson 2007:

- Divide nodes of adjacency graph of K into two disjoint sets: A-nodes that correspond to diagonal entries of A and remaining nodes are C-nodes.
- A C-node can only be ordered after all its A-node neighbours K have been ordered.

Constrained orderings

- Provided A is definite and B is of full row rank, with this ordering can prove the LDL^T factorization exists, with L unit lower triangular and D diagonal.
- Moreover, the pivots associated with A-nodes are positive and those associated with C-nodes are negative.
- Rescaling, $L \leftarrow L|D|^{1/2}$ and $D \leftarrow sign(D) = diag(\pm 1)$

This gives a signed Cholesky factorization of K.

Is this a good approach?

- **Advantage:** simple modification to Cholesky code, avoiding need for pivoting (static data structures set up during analyse phase).
- **Disadvantages:**
  - no pivoting so stability not guaranteed
  - constrained ordering may lead to more fill (thus more work and more memory)

Our experiments found generally better to use a state-of-the-art indefinite direct solver (such as `HSL_MA97`) that allows sparsity-preserving ordering and incorporates threshold pivoting.

What about the incomplete case?

- For the incomplete case, having the best sparsity-preserving ordering may not be so important.
  A good ordering may be sufficient.
- Would like to be able to exploit incomplete Cholesky factorizations that have proved memory efficient and robust for SPD problems.

Incomplete Cholesky

Let's consider the case of a symmetric positive definite A.

Variants of incomplete Cholesky (IC) factorizations include:

- $IC(\tau)$: Dropping by value (Tuff and Jennings '73)
- $IC(\ell)$: Originally exploited finite difference-based structure (small number of sub-diagonals). Generalised to level-based approach to preserve structure (Watts '81)
- $IC(p)$: Limited/prescribed memory: eg. Axelsson, Munksgaard '83; Jones, Plassman '95; Saad '94.

Tismenetsky '91 approach

Tismenetsky's approach is based on decomposing A into the form

$$A = (L + R)(L + R)^T - E$$

- L is lower triangular with positive diagonal entries used for preconditioning,
- R is strictly lower triangular with small entries that is used to stabilise the factorization process, and
- E has the structure

$$E = RR^T.$$

Tismenetsky approach

- Consider first step of Gaussian elimination. Let $a_{2:n}$ denote the off-diagonal part of column 1 of A.
- Let $a_{2:n} = l_{2:n} + r_{2:n}$ where if $a_i \neq 0$ either $l_i = 0$ and $r_i = a_i$ or $l_i = a_i$ and $r_i = 0$.
- The Schur complement S is

$$S = A_{2:n,2:n} - a_{2:n} a_{2:n}^T$$

- That is,

$$S = A_{2:n,2:n} - (l_{2:n} + r_{2:n})(l_{2:n} + r_{2:n})^T$$

- Tismenetsky computes an incomplete factorization by not subtracting the term $r_{2:n} r_{2:n}^T$, using $l_{2:n}$ as first column of the incomplete factor L and not including $r_{2:n}$ in computed factorization.

Tismenetsky approach

- On j-th step, decompose col. 1 of Schur complement S into

$$l_j + r_j \quad \text{with} \quad |l_j|^T |r_j| = 0,$$

  where entries of l_j are retained in incomplete factorization and those in r_j are discarded.

- On next step, S updated by subtracting

$$(l_j + r_j)(l_j + r_j)^T.$$

- Tismenetsky omits the term

$$E_j = r_j r_j^T. \tag{1}$$

- Thus, symmetric positive semi definite matrix is implicitly added to A and factorization is guaranteed breakdown free.

Kaporin's use of drop tolerances

- Obvious choice for r_j are smallest off-diagonal entries in col j.
- Controls size of L but not memory required to compute it.
- Kaporin '98: entries of magnitude at least τ_1 kept in L and those smaller than τ_2 are dropped from R.
- Now E has structure

$$E = RR^T + F + F^T,$$

F strictly lower triangular matrix that is not computed;
R used in computation of L but discarded.

Problem of unrestricted L and R

Tismenetsky approach is robust and can achieve high quality preconditioner **but**

- memory demands prohibitively high;
- can be very expensive to compute.

Thus impractical for the very large problems iterative methods designed for.

Remedy: impose memory limit on L and R.

What about breakdown?

- If we impose memory limit and/or drop small entries, Tismenetsky approach not guaranteed breakdown free.
- We tried various approaches to overcome this (modifying small pivot Kershaw '78, Jennings-Malik approach ...)
- Found we obtained best results using a global diagonal shift (Manteuffel '80)

$$A \Leftarrow A - \alpha I$$

  Note: multiple restarts may be required.
- Ideas implemented in the software package `HSL_MI28`.

HSL_MI28

- User specifies the amount of fill allowed in each column of L and maximum number of entries in each column of R.
- The user may specify drop tolerances τ_1 and τ_2 for L and R.
- The matrix is optionally preordered and prescaled (both can significantly enhance performance).

Note: the widely-used ICFS code of Lin and Moré '99 implements a special case in which ordering is not incorporated, there is no dropping and $R = 0$.

HSL_MI28 results

- Very encouraging results for wide range of positive definite problems (see Scott and Tuma ACM TOMS 2014 and SISC 2014).
- In particular, found it is beneficial to use intermediate memory ($R \neq 0$) in the construction of L.
- To assess performance we define the efficiency of a preconditioner to be

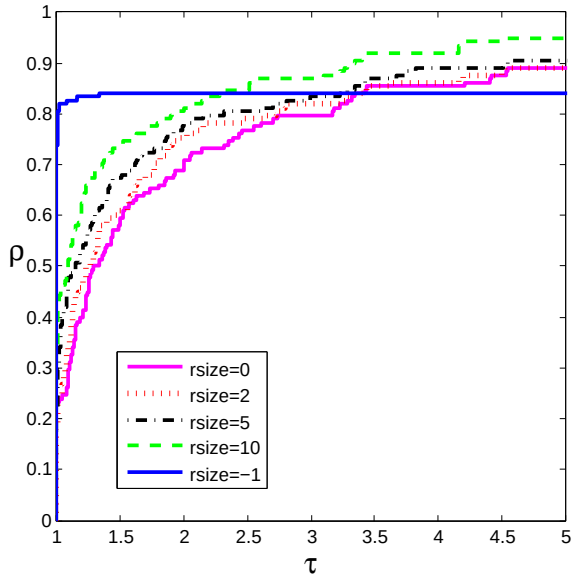
$$iter \times nz(L)$$

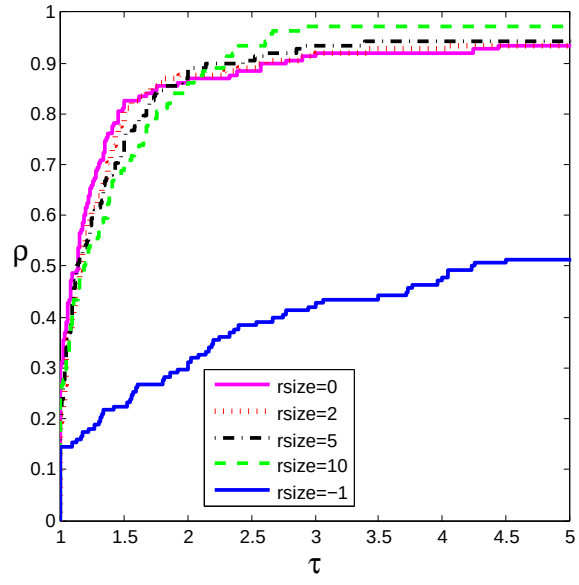
and use performance profiles (Moré, Dolan '02).

- Test set of 145 SPD problems from University of Florida Collection of order $n > 1000$.

Results for `rsize` varying

Efficiency (left) and total time (right) (`lsize`=5) for CG.





- `rsize`=`-1` is unlimited memory for R (not practical).

What about saddle-point systems?

Recall: we want an incomplete factorization preconditioner for saddle-point matrices

$$K = \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}$$

What about signed *IC*?

Challenge: can we adapt our *IC* code to compute a signed *IC* factorization preconditioner for K?

Two basic changes:

- Identify C-nodes and post process the pivot order so C-node is ordered only after all its A-node neighbours (constrained ordering).
- Allow use of two shifts and thus we factorize

$$\overline{K} = SQ \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} Q^T S + \begin{pmatrix} \alpha(1)I & 0 \\ & -\alpha(2)I \end{pmatrix}$$

where S is diagonal scaling, Q permutation.

Results $C = 0$

GMRES(100) convergence results. $\alpha(1) = 0.0$.
$fill_{IC} = nz(L)/nz(K)$

Identifier	*lsize*	*rsize*	$\alpha(2)$	$fill_{IC}$	iters
boyd1	20	20	0.0	0.82	7
d_pretok	5	5	2.56×10^{-1}	1.96	28
ncvxqp9	10	10	4.19×10^{3}	3.98	2
sit100	20	20	0.0	3.61	14
stokes64	10	10	0.0	2.73	157
tuma1	20	20	0.0	5.79	20
turon_m	20	20	1.60×10^{-2}	4.36	56

Results $C = 10^{-8}$ (interior-point matrices)

GMRES(100) convergence results. *lsize* = *rsize* = 10.

Identifier	$\alpha(1)$	$\alpha(2)$	$fill_{IC}$	iters
c-55	0.01	0.64	2.08	117
c-68	0.01	2.56	2.31	44
c-69	0.01	0.64	2.35	70
c-70	0.01	0.64	2.32	72
c-71	0.01	0.04	2.17	83
c-big	0.01	0.64	2.63	109

Some comparisons with SYM_ILDL

SYM_ILDL is code by Greif, He and Liu (2013).

- Based on the earlier work by Li and Saad (2005).
- Performs an incomplete factorization of sparse symmetric indefinite matrices with Bunch-Kaufman pivoting for numerical stability and to prevent breakdown.
- Incomplete factorization LDL^T, where D is block diagonal, with blocks of order 1 and 2, corresponding to 1×1 and 2×2 pivots.
- User parameters *fill* and *tol* control $nz(L)$.

Some comparisons with SYM_ILDL

GMRES(100) results with $lsize = rsize = 30$ and $\alpha_{in}(1:2) = 0.01$.
SYM-ILDL is run with $fill = 12.0$ and $tol = 0.003$.

Identifier	Signed *IC*				SYM-ILDL	
	$\alpha_{out}(1)$	$\alpha_{out}(2)$	$fill_{IC}$	iters	$fill_{IC}$	iters
GHS_indef/c-55	0.01	0.08	3.37	41	4.35	78
GHS_indef/c-68	0.01	0.01	4.06	14	4.31	61
GHS_indef/c-69	0.0025	0.01	4.00	30	3.81	42
GHS_indef/c-70	0.01	0.01	3.88	45	3.61	22
GHS_indef/c-71	0.01	0.01	3.51	53	4.03	37

Future directions

- Results suggest signed *IC* approach can provide good preconditioners for saddle-point systems.
  But can we do better?
- Currently developing limited memory incomplete LDL^T factorization code for general indefinite systems, again using intermediate memory that is discarded.
- Also exploring effects of matching-based orderings (used by Hagemann and Schenk '06 for incomplete factorizations).

Thank you!

HSL_MI28 (incomplete Cholesky) and HSL_MI30 (signed incomplete Cholesky for saddle-point systems) are available as part of HSL 2013. Free for academic use so please try and feedback!

MATLAB interfaces are available `http://www.hsl.rl.ac.uk`

More details and results: see Technical Report RAL-P-2014-003

Supported by EPSRC grant EP/I013067/1

Grant Agency of the Czech Republic Project No. P201/13-06684