On positive semidefinite modification schemes for incomplete Cholesky factorization

Jennifer Scott

STFC Rutherford Appleton Laboratory

Miroslav Tůma

Institute of Computer Science
Academy of Sciences of the Czech Republic

Preconditioning 2013, St Anne's College, Oxford

Introduction

We are interested in the efficient and robust solution of large sparse symmetric linear systems

$$Ax = b, \ A \in R^{n \times n}$$

In this talk, we focus on Incomplete Cholesky (IC) factorizations

$$A \simeq LL^T$$

used with the conjugate gradient (CG) method.

Incomplete factorization: some entries that occur in complete factorization are ignored.

Introduction

- Long history of incomplete factorizations.
- Early days (late 1950s and 1960s) motivated by finite differences for PDEs. Often for specific problems.
- Real revolution in practical use and growth in popularity came in late 1970s.
- In particular, Meijerink and van der Vorst '77 recognised potential of incomplete factorizations as preconditioners for use with CG and proved existence for M-matrices (later extended to H-matrices).

Introduction

Different variants of incomplete factorizations:

- $IC(\tau)$: 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: Axelsson, Munksgaard '83; Jones, Plassman '95; Saad '94.

Lots of variations/hybrids that combine approaches.

Introduction: problem of breakdown

- Kershaw '78 locally perturbed zero or negative diagonal entries to prevent breakdown so method more widely applicable. Straightforward but can give large growth and unstable preconditioner.
- Manteuffel '80 proposed global diagonal shift so that $A + \alpha I$ factorized for some $\alpha > 0$. Shift α chosen by trial-and-error but can be effective.
- Alternative approach: positive semi-definite modifications.

Our goals

- Study two positive semi-definite modification schemes:
  - Jennings and Malik '77,'78 (and Ajiz and Jennings '84)
  - Tismenetsky '91 (and Kaporin '98)

Our goals

- Study two positive semi-definite modification schemes:
  - Jennings and Malik '77,'78 (and Ajiz and Jennings '84)
  - Tismenetsky '91 (and Kaporin '98)
- Seek to gain better understanding and to explore the relationship between them.

Our goals

- Study two positive semi-definite modification schemes:
  - Jennings and Malik '77,'78 (and Ajiz and Jennings '84)
  - Tismenetsky '91 (and Kaporin '98)
- Seek to gain better understanding and to explore the relationship between them.
- Propose memory-efficient variant of Tismenetsky approach, optionally combined with Jennings and Malik modifications or diagonal shifts.

Our goals

- Study two positive semi-definite modification schemes:
  - Jennings and Malik '77,'78 (and Ajiz and Jennings '84)
  - Tismenetsky '91 (and Kaporin '98)
- Seek to gain better understanding and to explore the relationship between them.
- Propose memory-efficient variant of Tismenetsky approach, optionally combined with Jennings and Malik modifications or diagonal shifts.
- Present comprehensive numerical results.

Positive semi-definite modifications I

- Diagonal modification scheme first introduced by Jennings and Malik '77, '78 (also Jennings and Ajiz '84).
- Every time off-diagonal entry discarded, corresponding diagonal entries modified by adding SPSD matrix

$$\begin{array}{cc} & \begin{array}{ccccc} \quad & i & & j & \quad \end{array} \\ \begin{array}{c} \\ i \\ \\ j \\ \\ \end{array} & \begin{pmatrix} \ddots & & & & \\ & |a_{ij}| & & -|a_{ij}| & \\ & & \ddots & & \\ & -|a_{ij}| & & |a_{ij}| & \\ & & & & \ddots \end{pmatrix} \end{array}$$

Jennings-Malik approach

- Breakdown-free factorization that can be expressed as

$$A = LL^T + E$$

  where error matrix E is sum of SPSD matrices.
- But modifications to A can be significant.
- Popular in some engineering applications.

Positive semi-definite modifications II

- More sophisticated modification scheme due to Tismenetsky '91 (and Kaporin '98).
- Introduces use of intermediate memory that is employed during construction of L but then discarded.
- Shown to be very robust but it "has unfortunately attracted surprisingly little attention" (Benzi '02).
- One possible reason for this is it suffers from a serious drawback: memory requirements can be prohibitively high.

We aim to address memory problem, while retaining robustness.

Tismenetsky approach

Based on matrix decomposition of form

$$A = LL^T + LR^T + RL^T + \hat{E}$$

- 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
- $\hat{E}$ has the structure
$$\hat{E} = RR^T.$$

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

$$\hat{E}_j = r_j r_j^T. \tag{1}$$

- Thus, SPSD matrix implicitly added to A.

Can we compare the two approaches?

- Standard tool in modified IC (Gill, Murray, Wright '81, survey by Fang, O'Leary '08): consider norm of error matrix $E = A - LL^T$.
- Jennings-Malik implies a smaller $\| E \|$:

Theorem (Scott and Tůma)
At stage j, assume S has been computed and its first column split into l_j and r_j. Then the 2-norm of the Jennings-Malik modification that compensates for all the dropped entries is not larger than the 2-norm of the Tismenetsky modification corresponding to adding $r_j r_j^T$ to the corresponding positions.

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 $\hat{E}$ has structure

$$\hat{E} = RR^T + F + F^T,$$

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

Problem of unrestricted L and R

- With no restriction on size of L and R, can achieve high quality preconditioner but memory demands high.
- Also can be very expensive to compute making approach 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.
- Use global diagonal shift? (Manteuffel) Note: multiple restarts may be required so potentially expensive.
- Or combine with Jennings-Malik compensation?

How to combine approaches?

There are a number of possibilities:

- Compensate for all entries not retained in L or R.
- Allow entries in RR^T that do not lead to any further fill-in and compensate for all remaining entries of RR^T.

Test environment

- Problems from University of Florida Collection.
- Selected all non-diagonal SPD matrices with $n > 1000$.
- Removed those with duplicate sparsity patterns.
- All problems prescaled (this is important).
- Following initial experiments, 8 problems discarded as unable to achieve convergence without large amount of fill.
- Test set of 145 problems.

Test environment (continued)

- CG used with $x_0 = 0$, b computed so that $x = 1$, and stopping criteria

  $$\|Ax_k - b\| \leq 10^{-10}\|b\|$$

  with limit of 2000 iterations.
- All software written in Fortran.

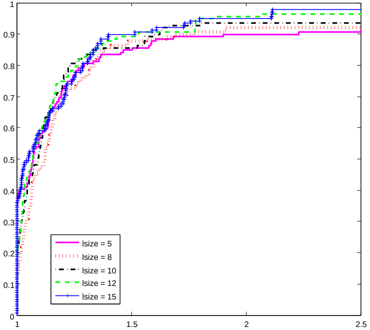
Test environment (continued)

- What to measure? iteration counts? timings? sparsity of L?
- We define the efficiency of preconditioner to be

$$iter \times nz(L)$$

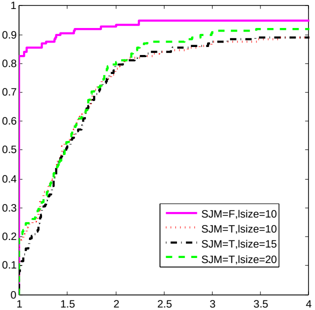
- Performance profiles (Moré, Dolan '02) used to assess performance.
- In our tests, `lsize` is max. number of fill entries in each col. of L and `rsize` is max. number of entries in each col. of R.

Efficiency for `rsize`=0, no diagonal compensation



- These results are without diagonal compensation and no dropping of small entries .... equilavent to ICFS code of Lin and Moré '99.
- Rather insensitive to choice of `lsize`.

Efficiency for `rsize`=0, with/without SJM



- These results are with and without standard Jennings-Malik (SJM) diagonal compensation.
- Conclude that compensation not generally useful in this case.

Iterations and time for `rsize`=0, with/without SJM

Comparison of using global diagonal shifts (GDS) with the Jennings-Malik strategy (SJM) (`lsize` = 10).
Figures in parentheses are number of shifts and final shift; times are in seconds.

Problem	Iterations		Total time	
	GDS	SJM	GDS	SJM
HB/bcsstk28	232 (2, $4.0 * 10^{-3}$)	468	0.120	0.221
Cylshell/s3rmq4m1	648 (2, $4.0 * 10^{-3}$)	838	0.381	0.459
GHS_psdef/ldoor	437 (3, $8.0 * 10^{-3}$)	643	66.4	91.5
GHS_psdef/audikw_1	707 (2, $2.0 * 10^{-3}$)	1442	157	303

- Our experience: generally better to use diagonal shift.

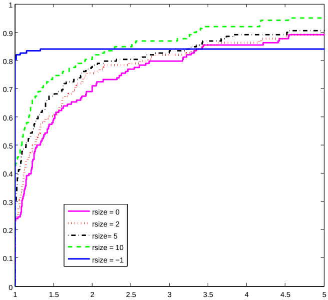
Results for `rsize` varying

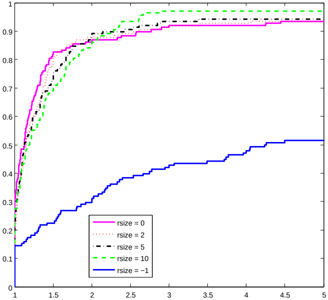
We now consider using intermediate memory (`rsize`$>$0).

We start by performing no diagonal compensation.

Results for `rsize` varying

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





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

Results with/without diagonal compensation

Recall:

Limited memory Tismenetsky approach based on decomposition

$$A = LL^T + LR^T + L^T R + \hat{E}, \qquad \hat{E} = RR^T + F + F^T,$$

where F is not computed but R is.

Results with/without diagonal compensation

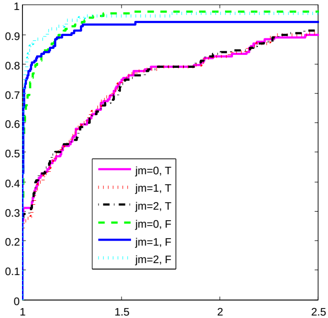
Consider three strategies for dealing with RR^T:

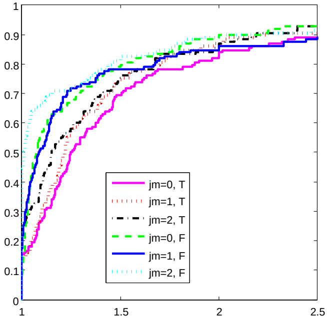
- `jm = 0`: allow entries of RR^T that cause no further fill in $LL^T + LR^T + L^T R$ and discard all other entries of RR^T.
- `jm = 1`: as above but use Jennings-Malik compensation for discarded entries of RR^T.
- `jm = 2`: discard all entries of RR^T.

We run these options with (T) and without (F) diagonal compensation for entries discarded from R.

Results with/without diagonal compensation

Efficiency (left) and total time (right) (`lsize=rsize=10`)





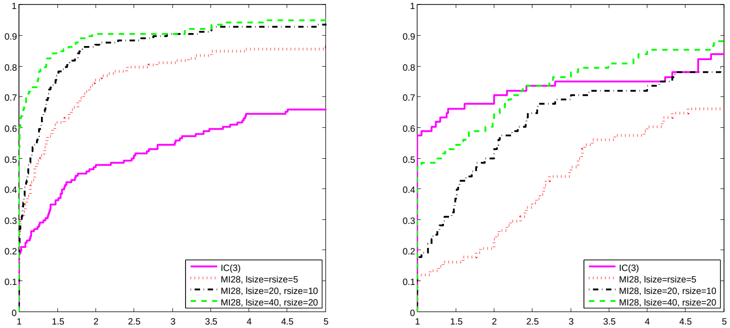
- Compensating for dropped entries of R generally not beneficial.
- Reliability slightly improved if entries of RR^T allowed (`jm=0`) but faster and better efficiency to ignore RR^T (`jm=2`).

New *IC* code

- Based on our findings, we have developed a new IC code called `HSL_MI28`.
- Can be used as a "black-box" to compute an efficient and robust *IC* preconditioner.
- But also flexible, allowing user to choose the scaling, ordering, diagonal shift, drop tolerances etc.
- Importantly, the amount of memory used (for both L and R) is under the user's control.

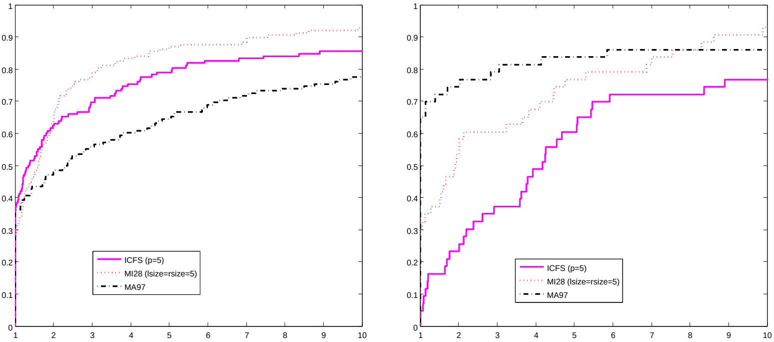
Comparison with level-based approach ($IC(3)$)

Efficiency (left) and iterations (right).



Comparison with direct solver HSL_MA97

Total time: all problems (left) and large problems (right).



HSL_MI28 can sometimes compete with direct solver
(and succeeds when HSL_MA97 runs out of memory).

Concluding remarks

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.
- The proposed limited memory Tismenetsky approach has been shown to be robust and efficient.

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.
- The proposed limited memory Tismenetsky approach has been shown to be robust and efficient.
- Using restricted intermediate memory improves efficiency.

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.
- The proposed limited memory Tismenetsky approach has been shown to be robust and efficient.
- Using restricted intermediate memory improves efficiency.
- But diagonal compensation to prevent breakdown appears less important than generally supposed.

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.
- The proposed limited memory Tismenetsky approach has been shown to be robust and efficient.
- Using restricted intermediate memory improves efficiency.
- But diagonal compensation to prevent breakdown appears less important than generally supposed.
- Our extensive experiments favour use of global diagonal shifts (works well provided the problem is well scaled).

Concluding remarks

- We have explored the use of diagonal compensation with a limited memory Tismenetsky approach.
- The proposed limited memory Tismenetsky approach has been shown to be robust and efficient.
- Using restricted intermediate memory improves efficiency.
- But diagonal compensation to prevent breakdown appears less important than generally supposed.
- Our extensive experiments favour use of global diagonal shifts (works well provided the problem is well scaled).
- New IC code `HSL_MI28`.

Thank you!

HSL_MI28 is available (without charge) as part of HSL 2013.

Technical Reports RAL-P-2013-004 and RAL-P-2013-005.

Supported by EPSRC grant EP/I013067/1
Grant Agency of the Czech Republic Project No. P201/13-06684