The challenge of large sparse rank deficient least squares problems

Jennifer Scott

STFC Rutherford Appleton Laboratory

With thanks to Nick Gould and Miroslav Tůma

Sparse Days at CERFACS, 30th June – 1st July 2016

Introduction

Least squares are used across a wide range of disciplines:

- simple curve fitting
- estimation of satellite image sensor characteristics
- powering internet mapping services
- exploration seismology
- NMR spectroscopy
- ultrasound for medical imaging
- aerospace systems
- neural networks
- data assimilation for weather forecasting and for climate modelling

Introduction

Linear least squares (LS) arises on its own and as subproblem of nonlinear least squares problem.
Solve

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|Ax - b\|_2,$$

where $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ is large and sparse and $b \in \mathbb{R}^m$.

Focus today is on A rank deficient and our interest is in using Cholesky-type factorizations and employing regularization.

These problems are **tough** and we struggle to solve them ...

LS problem is mathematically equivalent to solving $n \times n$ *normal equations*

$$Cx = A^T b, \qquad C = A^T A.$$

If A is full rank, C is positive definite so use sparse Cholesky solver

- CHOLMOD (Davis 2008),
- HSL_MA87 (Hogg, Reid and Scott 2010).

LS problem is mathematically equivalent to solving $n \times n$ *normal equations*

$$Cx = A^T b, \qquad C = A^T A.$$

If A is full rank, C is positive definite so use sparse Cholesky solver

Compute factorization

$$C = LL^T,$$

where L is lower triangular.

Solve $Lz = A^T b$ then $L^T x = z$.

Note: in practice, scale C to improve stability and preorder to keep L as sparse as possible.

Normal equations

Condition number of C is the square of the condition number of A so normal equations can be highly ill-conditioned.

If A does not have full column rank, then C is positive semidefinite so Cholesky factorization suffers breakdown (zero or negative pivot encountered).

Obvious remedy: use a general sparse symmetric solver that employs pivoting.

But we would really like to use a Cholesky solver as it is simpler and has greater potential for parallelism.

Idea: use regularization and then an iterative method.

Normal equations: regularization

Choose a shift $\alpha > 0$ and compute Cholesky factorization of

$$C_\alpha = SA^T AS + \alpha I.$$

where S is a diagonal scaling matrix.

α is also referred to as a *Tikhonov regularization* parameter.

If α is large enough then C_α is positive definite and factorization will not break down.

Potential problem: if the regularized scaled normal equations

$$C_\alpha x_\alpha = S(C + S^{-1}\alpha I S^{-1})S x_\alpha = SA^T b$$

are solved, the computed value of least squares objective

$$\|r_\alpha\|_2 = \|b - ASx_\alpha\|_2$$

may differ from the optimum for the original problem.

Solution: Seek to recover x by applying a refinement process.

The standard approach for linear systems is to employ a small number of steps of iterative refinement.

However, iterative refinement not effective when applied to normal equations if C is ill conditioned (Björk '96).

Instead, we propose using Cholesky factors $L_\alpha L_\alpha^T$ of C_α as a preconditioner for LSMR applied to original (unshifted) problem.

What is LSMR?

Want to apply iterative method to normal equations without explicitly forming normal matrix $C = A^T A$ (expensive, dense, inaccurate...).

What is LSMR?

Want to apply iterative method to normal equations without explicitly forming normal matrix $C = A^T A$ (expensive, dense, inaccurate...).

CGLS or CGNR (Hestenes and Stiefel 1952) is equivalent to conjugate gradients (CG) applied to C.

What is LSMR?

Want to apply iterative method to normal equations without explicitly forming normal matrix $C = A^T A$ (expensive, dense, inaccurate...).

CGLS or CGNR (Hestenes and Stiefel 1952) is equivalent to conjugate gradients (CG) applied to C.

LSQR (Paige and Saunders 1982) is based on Golub-Kahan bidiagonalization of A. Aims to improve numerical stability.

What is LSMR?

Want to apply iterative method to normal equations without explicitly forming normal matrix $C = A^T A$ (expensive, dense, inaccurate...).

CGLS or CGNR (Hestenes and Stiefel 1952) is equivalent to conjugate gradients (CG) applied to C.

LSQR (Paige and Saunders 1982) is based on Golub-Kahan bidiagonalization of A. Aims to improve numerical stability.

LSMR (Fong and Saunders 2001) is also based on Golub-Kahan. But is equivalent to MINRES (Paige and Saunders 1975) applied to the normal equations.

What is LSMR?

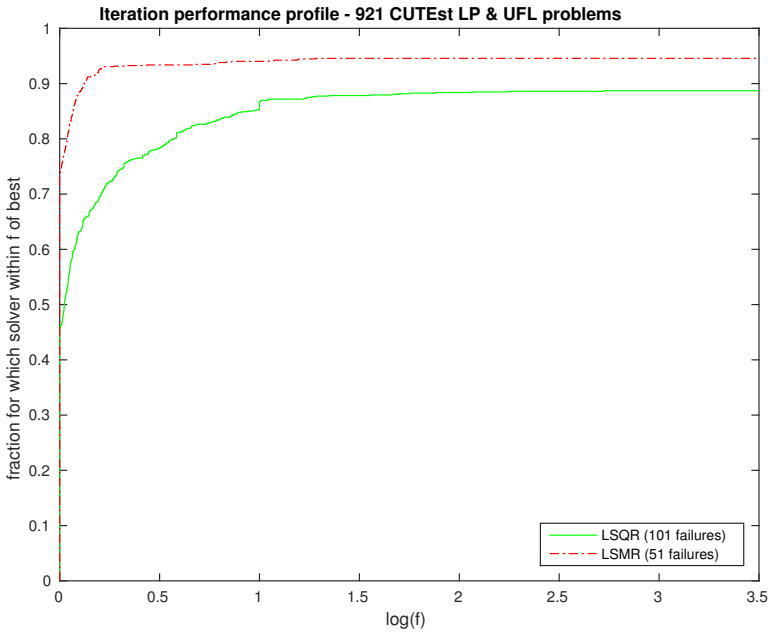
Want to apply iterative method to normal equations without explicitly forming normal matrix $C = A^T A$ (expensive, dense, inaccurate...).

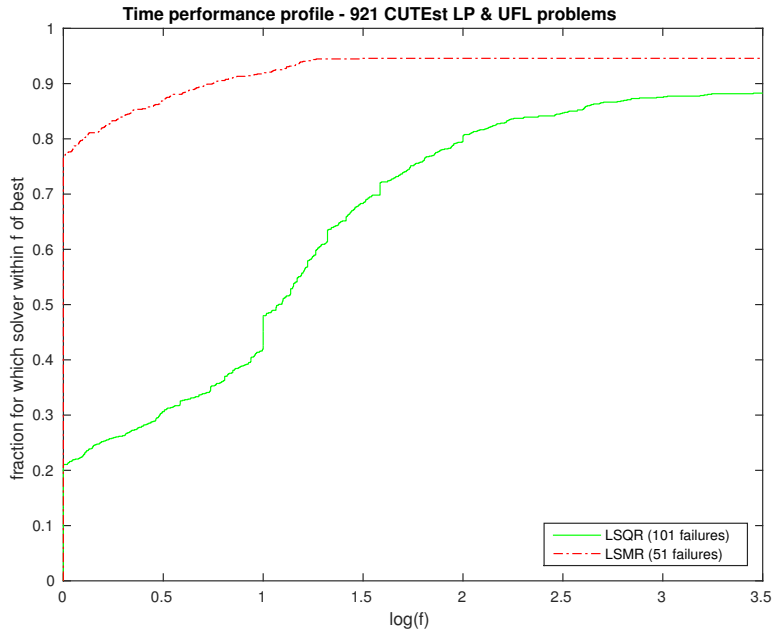
CGLS or CGNR (Hestenes and Stiefel 1952) is equivalent to conjugate gradients (CG) applied to C.

LSQR (Paige and Saunders 1982) is based on Golub-Kahan bidiagonalization of A. Aims to improve numerical stability.

LSMR (Fong and Saunders 2001) is also based on Golub-Kahan. But is equivalent to MINRES (Paige and Saunders 1975) applied to the normal equations.

For LSMR, $\|A^T r_k\|_2$ and $\|r_k\|_2$ decrease monotonically.
LSMR may be able to terminate significantly earlier than LSQR.





Iteration performance profile (left) and time performance profile (right) for LSMR and LSQR for our complete CUTEst and UFL test set of 921 eligible problems with no preconditioning.

How to choose α?

Recall: we propose using

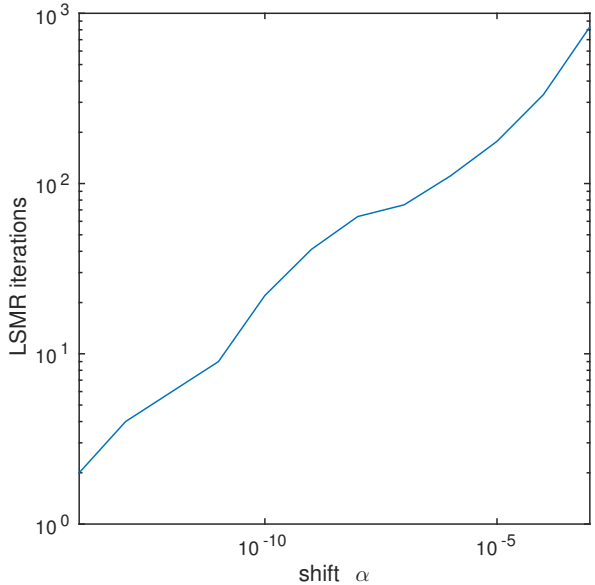
$$C_\alpha = SA^T AS + \alpha I.$$

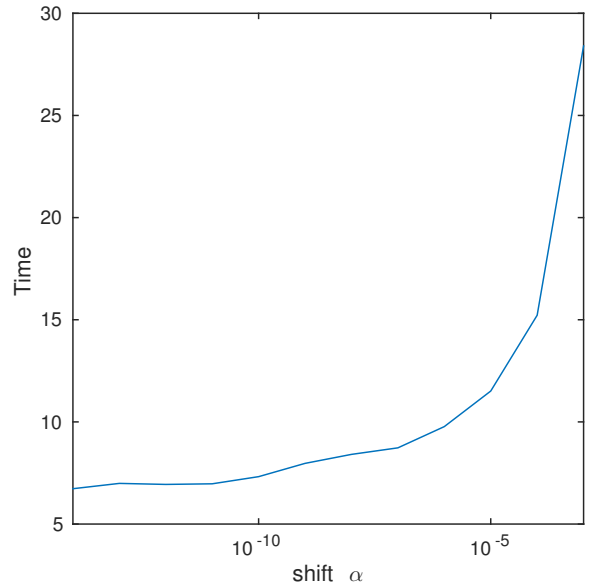
α should be small but large enough that C_α is positive definite.

Should be related to $\lambda_{min}(SA^T AS)$ but unavailable.

If initial choice is too small, restart factorization with larger α until breakdown is avoided.

In practice, found if we have scaled then can often choose very small α eg 10^{-14}.





The effect of the shift α on the number of LSMR iterations (left) and the time (right) for problem `Maragal_6`.

Comparison of using Cholesky code (HSL_MA87) to compute a preconditioner for LSMR ($\alpha = 10^{-14}$) with symmetric indefinite solver (HSL_MA97) applied to normal equations.

Problem	HSL_MA87			HSL_MA97	
	$nnz(L_\alpha)$	*time*	*iter.*	$nnz(L)$	*time*
BAXTER	6.83×10^6	0.32	24	6.95×10^6	0.44
208bit	8.60×10^7	6.62	1	8.57×10^7	25.5
Maragal_6	4.96×10^7	6.79	2	5.06×10^7	18.1
Maragal_8	8.85×10^7	7.42	26	9.18×10^7	23.8
mri2	3.43×10^7	1.99	1	3.79×10^7	5.90
tomographic1	2.96×10^7	0.70	2	3.27×10^7	2.58

Here and elsewhere, runs are on 4 processors, times are in seconds.

Alternative approach: augmented system

Least squares problem is also mathematically equivalent to solving the $(m+n) \times (m+n)$ *augmented system*

$$K \begin{bmatrix} \gamma^{-1} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \qquad K = \begin{bmatrix} \gamma I_m & A \\ A^T & 0 \end{bmatrix},$$

where $\gamma > 0$ and $r = b - Ax$ is the residual vector.

The computed factorization is

$$SKS = (PL)D(PL)^T,$$

with S a diagonal scaling matrix, P a permutation matrix, and D block diagonal (with 1×1 and 2×2 blocks).

Pivot order chosen using sparsity pattern.

Given a pivot threshold $u \in [0, 0.5]$, stability criteria:

- a 1×1 pivot on column k is stable if

$$u . \max_{i>k} |K_{i,k}^{(k)}| < |K_{k,k}^{(k)}|$$

- a 2×2 pivot on columns k and $k+1$ is stable if

$$u . \left| \begin{pmatrix} K_{k,k}^{(k)} & K_{k,k+1}^{(k)} \\ K_{k+1,k}^{(k)} & K_{k+1,k+1}^{(k)} \end{pmatrix}^{-1} \right| \begin{pmatrix} \max_{i>k+1} |K_{i,k}^{(k)}| \\ \max_{i>k+1} |K_{i,k+1}^{(k)}| \end{pmatrix} \leq \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The zero $(2,2)$ block in K can lead to many modifications being made to the pivot order; this is reflected in the number of delayed pivots reported by HSL_MA97 (*ndelay*).

HSL_MA97 for solving normal equations and augmented system.

Problem	Normal		Augmented	
	$nnz(L)$	*time*	$nnz(L)$	*time*
BAXTER	6.95×10^6	0.44	1.38×10^7	0.73
208bit	8.57×10^7	25.5	1.30×10^8	44.1
Maragal_6	5.06×10^7	18.1	2.30×10^7	3.02
Maragal_8	9.18×10^7	23.8	1.75×10^8	89.2
mri2	3.79×10^7	5.90	1.48×10^8	74.7
tomographic1	3.27×10^7	2.58	5.10×10^7	5.98

HSL_MA97 with threshold parameter $u = 0.01$ (default) and 10^{-8}.

Problem	$u = 0.01$			$u = 10^{-8}$		
	$nnz(L)$	*ndelay*	*time*	$nnz(L)$	*ndelay*	*time*
BAXTER	1.38×10^{7}	3.41×10^{4}	0.73	1.49×10^{6}	4.75×10^{3}	0.22
208bit	1.30×10^{8}	2.45×10^{4}	44.1	1.28×10^{8}	1.09×10^{4}	42.1
Maragal_6	2.30×10^{7}	4.18×10^{4}	3.02	2.30×10^{7}	4.17×10^{4}	2.75
Maragal_8	1.75×10^{8}	2.92×10^{5}	89.2	1.77×10^{8}	2.89×10^{5}	58.0
mri2	1.48×10^{8}	1.79×10^{5}	74.7	1.44×10^{8}	1.75×10^{5}	38.7
tomographic1	5.10×10^{7}	7.62×10^{4}	5.98	4.68×10^{7}	6.00×10^{4}	4.78

- Smaller u does not significantly reduce modifications to pivot order.
- Most of the time gain comes from faster factorization at root node with smaller u.

Regularized augmented system

$$K_\beta y_\beta = c, \quad K_\beta = \begin{bmatrix} I_m & A \\ A^T & -I_n\beta \end{bmatrix}, \quad y_\beta = \begin{bmatrix} r(x_\beta) \\ x_\beta \end{bmatrix}, \quad c = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

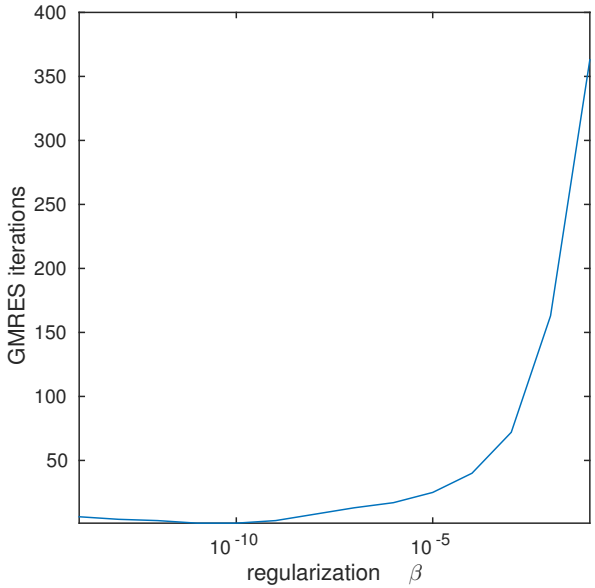
where $r(x_\beta) = b - Ax_\beta$ and $\beta > 0$.

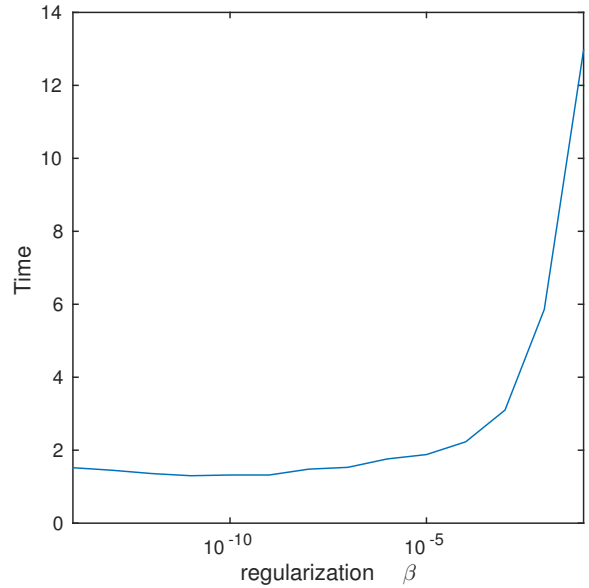
This system is symmetric quasi-definite (SQD).

Vanderbei (1995) showed SQD systems are strongly factorizable, i.e., a signed Cholesky factorization LDL^T (with D diagonal having both positive and negative entries) exists. But may be unstable.

Regularized augmented system

- Apply HSL_MA97 to K_β with $u = 0.0$.
- Uses 1×1 and 2×2 pivots.
- Use the computed factors as a preconditioner for GMRES applied to the original system ($\beta = 0$).





The effect of the regularization parameter β on the number of GMRES iterations (left) and the time (right) for problem `Maragal_6` (threshold $u = 0.0$).

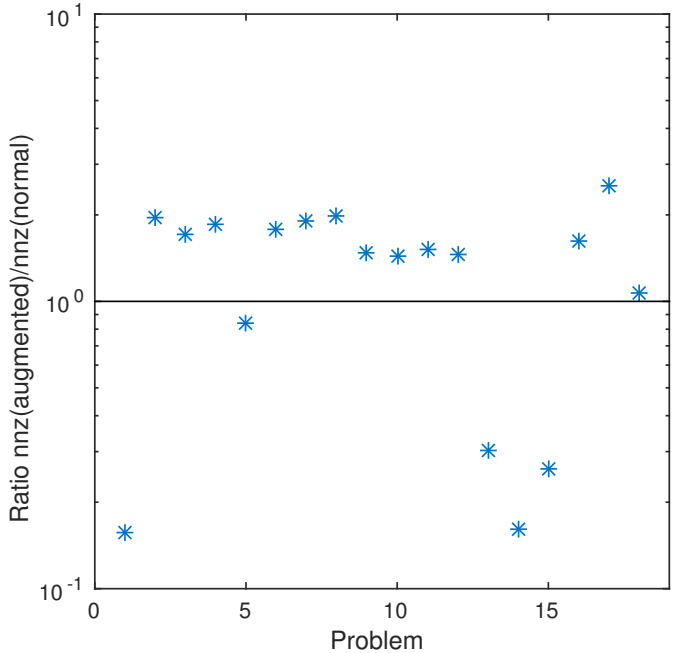
Results for solving the regularized augmented (SQD) system with $\beta = 10^{-8}$ using HSL_MA97 ($u = 0.0$) factors to precondition GMRES.

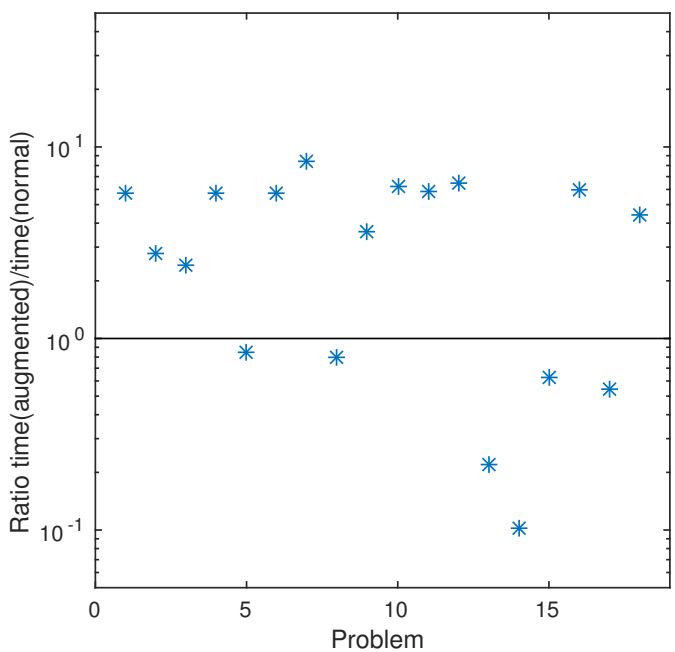
Number in parenthesis are for **no** regularization.

Problem	$nnz(L_\beta)$	$nnz(L)$	$time_f$	$time_{tot}$	$time$	$iter$
BAXTER	1.07×10^6	(1.49×10^6)	0.22	1.84	(0.22)	234
208bit	1.25×10^8	(1.28×10^8)	41.2	42.7	(42.1)	8
Maragal_6	1.50×10^7	(2.30×10^7)	1.28	1.50	(2.75)	8
Maragal_8	2.31×10^7	(1.77×10^8)	3.74	4.62	(58.0)	17
mri2	8.72×10^6	(1.44×10^8)	1.06	1.06	(38.7)	0
tomographic1	3.16×10^7	(4.68×10^7)	2.70	3.08	(4.78)	5

Regularized normal equations or augmented system?

Ratios of the number of entries in the factor (left) and the time (right) for the regularized augmented system solved using HSL_MA97 in indefinite mode and the shifted normal equations solved using the positive-definite solver HSL_MA87.





Findings for direct solvers

- Use the factors from the direct solver applied to regularized system as a preconditioner for an iterative solver.
- Regularization really helps both normal equations and augmented system.
- Typically only small number of iterations needed.
- **Conclude:** attractive approach with regularized normal equations (using Cholesky solver) often giving best results (time and sparsity).

Findings for direct solvers

- Use the factors from the direct solver applied to regularized system as a preconditioner for an iterative solver.
- Regularization really helps both normal equations and augmented system.
- Typically only small number of iterations needed.
- **Conclude:** attractive approach with regularized normal equations (using Cholesky solver) often giving best results (time and sparsity).

Now consider incomplete factorization preconditioners ...

Preconditioners for iterative solvers

- Can incomplete factorizations be used as robust and efficient preconditioners for rank deficient LS?
- Incomplete factorization of C or K?
- Considerable effort focused on incomplete Cholesky (IC) factorizations for symmetric positive definite systems.
- Little done until recently on incomplete factorizations of saddle point systems.

IC factorizations

$$C \approx \tilde{L}\tilde{L}^T$$

- $\tilde{L}$ is sparser than L but aims to retain important details of L.
- Many different IC variants have been proposed for SPD systems.
- We use a limited memory approach (Scott and Tůma 2014) which for LS problems is implemented within the package `HSL_MI35`.

Decompose C into the form

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

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

$$E = RR^T.$$

HSL_MI35: LS incomplete Cholesky code

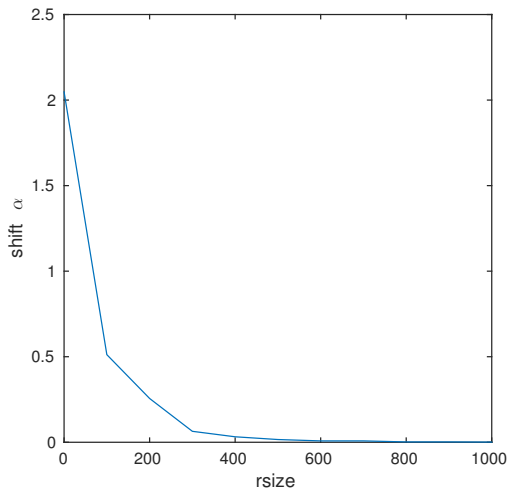
- User specifies the maximum number of entries allowed in each column of L and R (parameters *lsize* and *rsize*). Largest entries are included in L; next largest in R and remainder are discarded.
- User may specify drop tolerances τ_1 and τ_2 for L and R.
- Matrix is optionally preordered and prescaled (both can significantly enhance performance).
- Robustness through global diagonal shifts

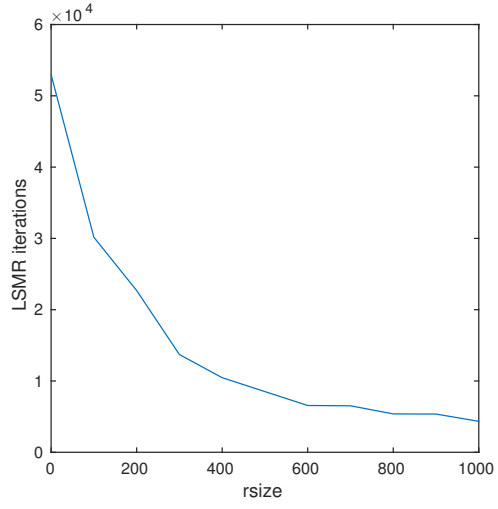
$$C \Leftarrow C + \alpha I.$$

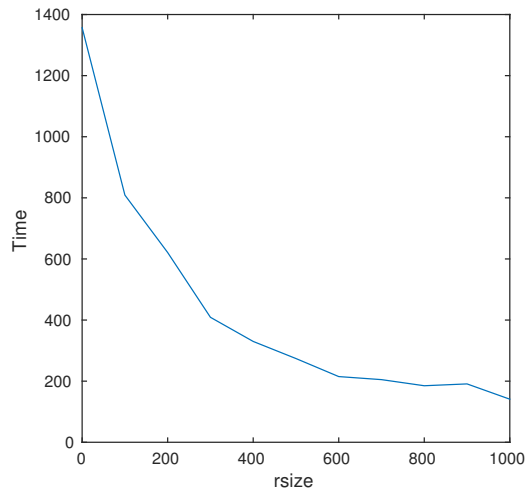
Use of α prevents breakdown and allows rank deficient problems to be tackled.

Potential benefits of employing intermediate memory R when constructing L.

Here *lsize* $= 200$ and *rsize* varies from 0 to 1000.







Plots are α (left), LSMR iterations (centre), time (right) for Maragal_8.

Results for LSMR with the IC factorization preconditioner from HSL_MI35 applied to $C = A^T A$.

Problem	*lsize*	*rsize*	$nnz(L)$	α	$time_f$	$time_{tot}$	*iter*
PDS-100	20	20	2.89×10^6	0.0	1.08	3.04	90
208bit	20	20	4.69×10^5	0.002	0.65	4.94	1430
Maragal_6	20	20	2.12×10^5	0.25	6.46	7.30	590
Maragal_8	200	1000	1.39×10^6	0.001	13.8	60.9	5120
mri2	20	20	7.83×10^5	2.05	3.17	19.0	2500
tomographic1	100	0	3.29×10^6	0.001	0.79	8.24	590

Preconditioning the augmented system

Recall

$$K = \begin{pmatrix} I_m & A \\ A^T & 0 \end{pmatrix}.$$

Two approaches to incomplete factorization

- Signed incomplete Cholesky factorization ie LDL^T with D diagonal with entries ± 1. This exploits block structure of K.
- General incomplete LDL^T factorization with D block diagonal. This ignores structure.

We consider signed IC as we have found this to be more efficient as well as more robust.

Signed IC factorizations

- Compute pivot order using standard sparsity-preserving algorithm (AMD, nested dissection ...) and then modify to allow for zero block.
  - Divide nodes of adjacency graph of K into two disjoint sets: A-nodes that correspond to diagonal entries of (1,1) block and remaining nodes are called C-nodes.
  - A C-node can only be ordered after all its A-node neighbours K have been ordered (constrained ordering).

Signed IC factorizations

- Compute pivot order using standard sparsity-preserving algorithm (AMD, nested dissection ...) and then modify to allow for zero block.

- Perform incomplete factorization of scaled and shifted system

$$\overline{K} = SQ \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} Q^T S + \begin{pmatrix} \alpha_1 I & 0 \\ & -\alpha_2 I \end{pmatrix}$$

where $\alpha_i \geq 0$, S is a diagonal scaling matrix, and Q holds pivot order.

Our code for this is `HSL_MI30`.

Results for MINRES with the signed IC factorization preconditioner from HSL_MI30 applied to the augmented system.

Problem	*lsize*	*rsize*	$nnz(L)$	α_2	$time_f$	$time_{tot}$	*iter*
PDS-100	20	0	5.21×10^6	0.0	1.51	2.46	24
208bit	50	50	1.53×10^6	0.001	2.95	6.98	683
Maragal_6	20	20	6.44×10^5	0.512	2.28	4.31	707
Maragal_8	100	0	3.68×10^6	0.016	1.94	30.9	1837
mri2	100	0	1.39×10^7	0.512	67.6	91.8	623
tomographic1	50	0	4.76×10^6	0.001	1.45	11.5	517

Here $\alpha_1 - 0$ (shift for (1,1) block).

Times in red indicate faster than we obtained for normal equations.

Currently, unable to predict whether to use normal equations or augmented system.

Concluding remarks

- This work forms part of a larger study into solving large-scale linear least squares problems (to appear in ACM TOMS).
- Other methods tested were generally worse when applied to rank deficient problems.
- We found some examples in UFL Collection that we were not able to solve with any of the non-commercial codes currently available.
- Thus there is a need for further work. Ideas???

Merci de votre attention!

HSL_MI30 and HSL_MI35 are available (free!) as part of HSL.

LSQR and LSMR available from Michael Saunders' web pages.

Reports on solving LS problems available from my web page.

Project supported by EPSRC grant EP/M025179/1