School of Mathematical, Physical
and Computational Sciences

University of Reading

Department of Mathematics and Statistics

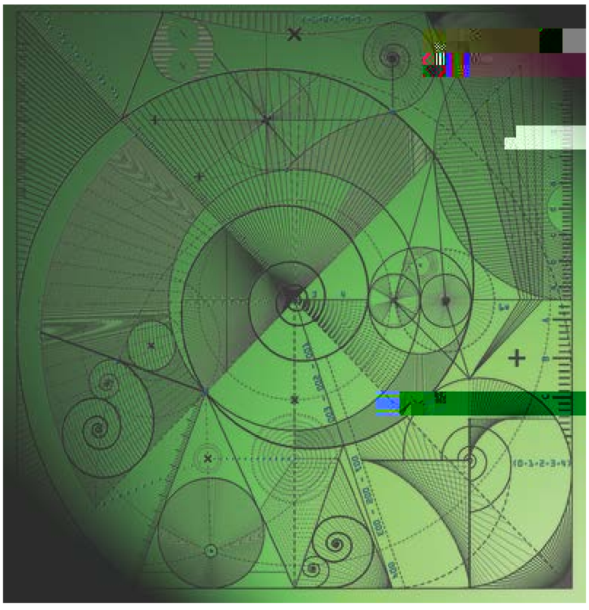
Preprint MPCS-2017-04

18 April 2017

A Schur complement approach to preconditioning sparse linear least-squares problems with some dense rows

by

Jennifer Scott and Miroslav Tůma



A SCHUR COMPLEMENT APPROACH TO PRECONDITIONING SPARSE LINEAR LEAST-SQUARES PROBLEMS WITH SOME DENSE ROWS

JENNIFER SCOTT AND MIROSLAV T UMA [y]

Abstract. The e ectiveness of sparse matrix techniques for directly solving large-scale linear least-squares problems is severely limited if the system matrix A has one or more nearly dense rows. In this paper, we partition the rows of A into sparse rows and dense rows (A_s and A_d) and apply the Schur complement approach. A potential di culty is that the reduced normal matrix A

In this paper, we exploit the fact that solving (1.4) is equivalent to solving the larger $(m+n)\times(m+n)$ augmented system

$$\begin{pmatrix} I_{m_s} & & A_s \\ & I_{m_d} & A_d \\ A_s^T & A_d^T & 0 \end{pmatrix} \begin{pmatrix} r_s \\ r_d \\ x \end{pmatrix} = \begin{pmatrix} b_s \\ b_d \\ 0 \end{pmatrix}; \tag{1.5}$$

where

$$r = \begin{pmatrix} r_s \\ r_d \end{pmatrix} = \begin{pmatrix} b_s \\ b_d \end{pmatrix} - \begin{pmatrix} A_s \\ A_d \end{pmatrix} x$$

is the residual vector. Here and elsewhere, I_k denotes the $k \times k$ identity matrix. The system (1.5) is symmetric indefinite and so, if there is sufficient memory available, a sparse direct solver that incorporates the use of numerical pivoting for stability can be used (well-known examples include MA57 [12] and HSL_MA97 [20] from the HSL mathematical software library [21], MUMPS [27] and WSMP [43]). Employing a general-purpose sparse solver ignores the block structure, although its use of a sparsity-preserving ordering (such original normal matrix C. Let $C_s = L_s L_s^T$ be the Cholesky factorization of C_s. Using this yields a block factorization

$$K =$$

A have full column rank and assume A_s has n_2 null columns with n_2 n. Assuming these columns are permuted to the end, we can split A into the form

$$A = \begin{matrix} A_1 & A_2 \end{matrix} \quad \begin{matrix} A_{s_1} & 0 \\ A_{d_1} & A_{d_2} \end{matrix} \tag{2.10}$$

with A_1 2 $R^{m \quad n_1}$ and A_2 2 $R^{m \quad n_2}$ $(n = n_1 + n_2)$. The following result from [39] shows that the solution of the least-squares problem can be expressed as a combination of partial solutions.

Lemma 2.1. Let the columns of A be split as in (2.10) and let z 2 R^{n_1} and W

Thus the computed value of the least-squares objective may di er from the optimum for the original problem. Having solved the regularized problem we want to recover the solution of the original problem. Following Scott [36], we propose doing this by using the factors of $K(\;)$ as a preconditioner for an iterative method applied to (2.1).

Let the Cholesky factorization of $C_s(\;)$ be $L_s(\;)L_s(\;)^T$. For $\;>\;0$, this is an approximate factorization of C_s, that is, $C_s \;\; L_s(\;)L_s(\;)^T$. More generally, let

$$C_s \quad \tilde{L}_s\tilde{L}_s^T; \tag{2.15}$$

where $\tilde{L}_s$ is lower triangular. We are interested in the case $\tilde{L}_s = L_s(\;)$ but our main focus is where $\tilde{L}_s$ is an incomplete Cholesky (IC) factor, that is, one that contains fewer entries than occur in a complete factorization. For very large systems, computing and factorizing C_s (or C_s

Many different IC factorizations have been proposed. Although they may be considered to be general purpose, most are best suited to solving particular classes of problems. For example, level-based methods are often most appropriate for systems with underlying structure, such as from finite element or finite difference applications. Here we use the limited memory based approach of Scott and Tůma [37, 38], that has been shown in [18] to result in effective preconditioners for a wide range of least-squares problems. The basic scheme employs a matrix factorization of the form

$$C_s \approx (\tilde{L}_s + R)(\tilde{L}_s + R)^T, \qquad (2.20)$$

where $\tilde{L}_s$ is the lower triangular matrix with positive diagonal entries that is used for preconditioning and R is a strictly lower triangular matrix with small entries that is used to stabilize the factorization process but is then discarded (it is not used as part of the preconditioner). The user specifies the maximum number of entries in each column of $\tilde{L}_s$ and R. At each step j of the incomplete factorization process, the largest entries are kept in column j of $\tilde{L}_s$, the next largest are kept in column j of R, and the remainder (the smallest entries) are dropped. In practice, C_s is optionally preordered and scaled and, if necessary, shifted to avoid breakdown of the factorization (which occurs if a non positive pivot is encountered) [25].

3. Numerical experiments.

In this section, we present numerical results to illustrate potential of the Schur complement approach and, in particular, demonstrate that it allows us to solve some problems that are intractable if dense rows are ignored. Results are included for direct solvers and for iterative solvers that can be used to solve very large problems.

3.1. Test environment.

The characteristics of the machine used to perform our tests are given in Table 3.1. All software is written in Fortran and all reported timings are elapsed times in seconds. In

Table 3.1
Test machine characteristics

CPU	Two Intel Xeon E5620 quadcore processors
Memory	24 GB
Compiler	gfortran version 4.8.4 with options -O3 -fopenmp
BLAS	Intel MKL

where $\begin{pmatrix} x^{(k)} \\ r_d^{(k)} \end{pmatrix}$ is the computed solution of (2.1) on the kth step. In our experiments, we set $\tilde{} = 10^{\ 7}$. With this choice, in most of our experiments (3.1) is satis ed with $= 10^{\ 6}$.

3.2. Test set 1. Our test problems are taken from the CUTEst linear programme set [17] and the UFL Sparse Matrix Collection [9]. In each case, the matrix is \cleaned" (duplicates are summed, out-of-range entries and explicit zeros are removed along with any null rows or columns). In our experiments, we use the following de nition for a dense row of A: given (0 < 1), row i of A is de ned to be dense if the percentage of entries in row i is at least .

Our rst test set is given in Table 3.2. The problems were chosen because they have at least one row that is more than 10% dense. They are also di cult problems to solve (see [18]); at least three of the problems are rank de cient. An estimate of the rank was computed by running the sparse symmetric inde nite solver HSL_MA97 on the augmented system (1.5) (with the pivot threshold parameter set to 0.5); for problems 12month1 and PDE1 there was insu cient memory to do this.

Table 3.2

Statistics for Test Set 1. m, n and $nnz(A)$ are the row and column counts and the number of nonzeros in A. nullity is the estimated de ciency in the rank of A, rdensity(A) is the largest ratio of number of nonzeros in a row of A to n over all rows, m_j ($j = 10; 20; 30; 40; 50$) is the number of rows of A with at least j% entries, and density(C) is the ratio of the number of entries in C to n^2. denotes insu cient memory to compute the statistic.

Problem	m	n	$nnz(A)$	nullity	rdensity(A)	m_{10}	m_{20}	m_{30}	m_{40}	m_{50}	density(C)
Trec14	15904	3159	2872265	0	0.791	2664	1232	649	346	150	$9.32\ 10^{\ 1}$
Maragal_6	21251	10144	537694	516	0.586	68	68	30	21	0	$7.49\ 10^{\ 1}$
Maragal_7	46845	26525	1200537	2046	0.360	85	43	21	0	0	$3.10\ 10^{\ 1}$
scsd8-2r	60550	8650	190210	0	0.100	40	0	0	0	0	$5.22\ 10^{\ 2}$
PDE1	271792	270595	990587	-	0.670	1	1	1	1	1	-
12month1	872622	12471	22624727	-	0.274	284	4	0	0	0	$6.87\ 10^{\ 1}$

Table 3.3

The e ects of varying the row density parameter on the number m_d of rows that are classed as dense and the density of C_s (the ratio of the number of entries in C_s to n^2).

Identi er	m	n		m_d	density (C_s)
Trec14	15904	3159	0.005	12643	$2.38\ 10^{\ 2}$
			0.010	9676	$8.52\ 10^{\ 2}$
			0.050	4467	$6.17\ 10^{\ 1}$
			0.100	2664	$8.31\ 10^{\ 1}$
Maragal_6	21251	10144	0.005	2923	$6.22\ 10^{\ 4}$
			0.010	823	$1.93\ 10^{\ 2}$
			0.100	68	$5.49\ 10^{\ 2}$
Maragal_7	46845	26525	0.001	4668	$2.15\ 10^{\ 4}$
			0.005	687	$9.02\ 10^{\ 3}$
			0.010	108	$1.70\ 10^{\ 2}$
			0.100	85	$1.78\ 10^{\ 2}$
scsd8-2r	60550	8650	0.050	50	$1.44\ 10^{\ 3}$
			0.100	40	$1.39\ 10^{\ 2}$
PDE1	271792	270595	0.660	1	$4.52\ 10^{\ 5}$
12month1	872622	12471	0.010	43951	$1.10\ 10^{\ 1}$

to m. For the Maragal problems, C_s is highly sparse if approximately 10% of the rows are classified as dense.

3.3. Test set 2. For our second test set, we take some of the CUTEst and UFL examples that do not initially contain dense rows and append some rows. This allows us to explore the effect of varying the number of dense rows as well as the density of these rows. The problems are listed in Table 3.4; these problems are all of full rank. When appending rows, the pattern of each such row is generated randomly with the requested density and the values of the entries are random numbers in $[1; 1]$.

For our solvers, the number of entries $nnz(C)$ in the normal matrix C can be at most huge(1) (2 10^9) where huge is the Fortran intrinsic function. If we add a single row with density $0{:}1$ to each of the matrices A_s in the lower part of Table 3.4 then $nnz(C)$ exceeds this limit. Thus for these examples and our current software, we cannot use any approach that requires the normal matrix to be computed.

Table 3.4

Statistics for Test Set 2. m_s, n and $nnz(A_s)$ are the row and column counts and the number of nonzeros in A_s. rdensity(A_s) is the largest ratio of number of nonzeros in a row of A_s to n over all rows, and density(C_s) is the ratio of the number of entries in C_s to n^2.

Problem	m_s	n	$nnz(A_s)$	rdensity(A_s)	density(C_s)
IG5-15	11369	6146	323509	1.95 10	

Table 4.1

Results for Test Set 1 of running the Cholesky direct solver HSLMA87 on the normal equations (without exploiting dense rows), using LSMR for refinement. nnz(L) denotes the number of entries in the Cholesky factor L of C and Its is the number of LSMR iterations. T_f, T_s and T_{total} denote the times (in seconds) to compute the normal matrix and factorize it, to run LSMR and the total time. denotes unable to form normal matrix C.

Identifier	m	n	nnz(L)	Its	T_f	T_s	T_{total}
Trec14	15904	3159	4.85×10^6	1	4.79	0.03	4.82
Maragal_6	21251	10144	4.96×10^7	3	13.1	0.12	13.2
Maragal_7	46845	26525	1.43×10^8	4	37.8	0.40	38.2
scsd8-2r	60550	8650	1.20×10^7	0	0.88	0.00	0.88
PDE1	271792	270595	-	-	-	-	-
12month1	872622	12471	7.27×10^7	1	42.5	0.35	42.9

Table 4.2

Results for Test Set 1 of solving the reduced augmented system (2.1) using the Schur complement approach and the Cholesky direct solver HSLMA87 is the row density parameter. density(C_s) is the ratio of the number of entries in the reduced normal matrix C_s to n^2, nnz(L) is total number of entries in the factors (that is, nnz$(L_s) + m_d(m_d+1)/2$), Its is the number of GMRES iterations. T_p, T_s and T) +

solving (2.18), and for forming and factorizing the Schur complement matrix (2.19). For problems Trec14, scsd8-2r and 12month1, results are given for more than one value of the parameter that controls which rows are classified as dense. As the density of C_s increases, a larger shift is needed to prevent breakdown of the IC factorization and this has the effect of decreasing the quality of the preconditioner. However, for small , for examples 12month1 and Trec14, m_d is large. Consequently, the factorization of the dense Schur complement $\tilde{S}$ is expensive and although the GMRES iteration count is much less than the LSMR count, for these two problems the Schur complement approach offers no significant benefit in terms of total time. For the other problems, exploiting the dense rows is advantageous. In particular, PDE1 could not be solved via the normal equations but the reduced augmented system approach performs well. We observe that for the rank deficient Maragal limit of 600 seconds. By contrast, for preconditioned GMRES on the reduced augmented system, the shift and the times to compute the incomplete factorization and achieve convergence are essentially independent of (and for this reason only results for = 1:0 are included in Table 5.4). Furthermore, this approach uses a smaller shift than for the normal equations and produces a much higher quality preconditioner, leading to signi cantly faster times. With more than one added row, the density of C often increases further making the normal equation approach even less feasible. For the augmented approach, adding more than one row does not a ect C_s or the time to compute the incomplete factorization but does result in the dense factorization of the Schur complement matrix becoming more expensive. For most of our test problems, the number of iterations decreases as the number of added rows increases (for example, pltexpA4 6, se0 and graphics but for others (including relat9), the converse is true (see Table 5.5).

Table 5.3

Results for Test Set 2 with a single dense row of density appended. Results are for preconditioned LMSR on the normal equations using the IC factorization preconditioner. denotes the global shift, IttheIt184augmeWIt18norrofationr.

Table 5.4

Results for Test Set 2 with a single dense row (= 1 :0) appended. Results are for preconditioned LMSR on the normal equations using the IC factorization preconditioner and for running GMRES on the reduced augmented system using the block IC factorization preconditioner.   denotes the global shift, Its is the number of iterations. T_p, T_s and T_{total} denote the times (in seconds) to compute the IC preconditioner, to run the iterative solver and the total time. { indicates statistic unavailable.

Problem	Normal equations with LSMR					Reduced augmented system with GMRES				
		Its	T_p	T_s	T_{total}		Its	T_p	T_s	T_{total}
IG5-15	$6.5536 \; 10^{1}$	810	0.92	0.82	1.73	1.024	337	0.47	1.08	1.55
psse0	$2.6214 \; 10^{2}$	33690	2.06	39.8	41.9					

REFERENCES

[1] E. D Andersen, J. Gondzio, C. Mészáros, and X. Xu. *Implementation of interior point methods for large scale linear programming*. HEC/Université de Genève, 1996.

[2] K. D. Andersen. A modified Schur complement method for handling dense columns in interior point methods for linear programming, 1996.

[3] O. D. Anderson. An improved approach to inverting the autocovariance matrix of a general mixed autoregressive moving average time process. *Australian J. Statistics*, 18(1-2):73{75, 1976.

[4] O. D. Anderson. On the inverse of the autocovariance matrix for a general moving average process. *Biometrika*, 63(2):391{394, 1976.

[5] H. Avron, E. Ng, and S. Toledo. Using perturbed QR factorizations to solve linear least-squares problems. *SIAM J. on Matrix Analysis and Applications*, 31(2):674{693, 2009.

[6] M. Benzi. Preconditioning techniques for large linear systems: a survey. *J. of Computational Physics*, 182(2):418{477, 2002.

[7] A. Björck. A general updating algorithm for constrained linear least squares problems. *SIAM J. on Scientific and Statistical Computing*, 5(2):394{402, 1984.

[8] A. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Philadelphia, 1996.

[9] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1{28, 2011.

[10] P. S. R. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. Springer, 4th ed. 2013 edition, 2012.

[11] H. S. Dollar and J. A. Scott. A note on fast approximate minimum degree orderings for matrices with some dense rows. *Numerical Linear Algebra with Applications*, 17:43{55, 2010.

[12] I. S. Duff. MA57{ a new code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30:118{154, 2004.

[13] D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM J. on Scientific Computing*, 33(5):2950{2971, 2011.

[14] A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra and its Applications*, 34:69{83, 1980.

[15] P. E. Gill, W. Murray, D. B. Ponceleon, and M. A. Saunders. Solving reduced KKT systems in barrier methods for linear and quadratic programming. Technical Report SOL 91-7, Department of Operations Research, Stanford University, 1991.

[16]

[30] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Transactions on Mathematical Software, 8(1):43–71, 1982.

[31] J. K. Reid and J. A. Scott. An out-of-core sparse Cholesky solver. ACM Transactions on Mathematical Software, 36(2):9:1–9:33, 2009.

[32] Y. Saad. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[33] Y. Saad and M. H. Schulz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7:856–869, 1986.

[34] M. A. Saunders. Cholesky-based methods for sparse least squares: The benefits of regularization. Technical Report SOL 95-1, Department of Operations Research, Stanford University, 1995. In L. Adams and J. L. Nazareth (eds.), Linear and Nonlinear Conjugate Gradient-Related Methods, SIAM, Philadelphia, 92–100 (1996).

[35] A. H. Sayed. Fundamentals of adaptive filtering. Wiley, 2003.

[36] J. A. Scott. On using Cholesky-based factorizations for solving rank-deficient sparse linear least-squares problems. Technical Report RAL-P-2016-005, Rutherford Appleton Laboratory, 2016.

[37] J. A. Scott and M. Tůma. HSL_MI28: an efficient and robust limited-memory incomplete Cholesky factorization code. ACM Transactions on Mathematical Software, 40(4):Art. 24, 19 pages, 2014.

[38] J. A. Scott and M. Tůma. On positive semidefinite modification schemes for incomplete Cholesky factorization. SIAM J. on Scientific Computing, 36(2):A609–A633, 2014.

[39] J. A. Scott and M. Tůma. Solving mixed sparse-dense linear least squares by preconditioned iterative methods. Technical Report RAL-TR-2017-P-001, Rutherford Appleton Laboratory, 2017.

[40] C. Sun. Dealing with dense rows in the solution of sparse linear least squares problems. Research Report CTC95TR227, Advanced Computing Research Institute, Cornell Theory Center; Cornell University, 1995.

[41] C. Sun. Parallel solution of sparse linear least squares problems on distributed-memory multiprocessors. Parallel Computing, 23(13):2075–2093, 1997.

[42] R. J. Vanderbei. Splitting dense columns in sparse linear systems. Linear Algebra and its Applications, 152:107–117, 1991.

[43] WSMP. Watson sparse matrix package (WSMP), 2016. http://researcher.watson.ibm.com/researcher/view_group.php?id=1426.