Science & Technology Facilities Council

Technical Report
RAL-TR-2014-011

Experiments using incomplete Cholesky factorization preconditioners for saddle-point systems arising in interior-point methods

J Scott

September 2014

Enquiries concerning this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358-6254

Experiments using incomplete Cholesky factorization preconditioners for saddle-point systems arising in interior-point methods

Jennifer Scott[1]

ABSTRACT

In the last couple of years there has been renewed interest in using a 3×3 block formulation of the symmetric indefinite sparse linear systems that arise from interior-point methods for quadratic optimization. This report presents a comparative study of factorizing the 2×2 and 3×3 block forms within an interior-point solver. We consider a sparse direct solver and then focus on using a new signed incomplete Cholesky factorization as a preconditioner for an iterative solver. The results confirm that the smaller 2×2 formulation should be used for the direct solver but for the iterative method, there is no conclusive winner.

Keywords: sparse matrices, sparse linear systems, indefinite symmetric systems, saddle-point systems, interior-point methods, iterative solvers, preconditioning, incomplete Cholesky factorization.

AMS(MOS) subject classifications: 65F05, 65F50

---

[1] Scientific Computing Department, Rutherford Appleton Laboratory, Harwell Oxford, Oxfordshire, OX11 0QX, UK.
Correspondence to: jennifer.scott@stfc.ac.uk
Supported by EPSRC grant EP/I013067/1.

September 25, 2014

1 Introduction

Consider the primal-dual pair of quadratic programs in standard form

$$\min_x c^T x + \frac{1}{2} x^T H x \quad \text{subject to } Jx = b, \quad x \geq 0, \tag{1.1}$$

$$\max_{x,y,z} b^T y - \frac{1}{2} x^T H x \quad \text{subject to } J^T y + z - Hx = c, \quad z \geq 0, \tag{1.2}$$

where the Hessian $H \in \mathcal{R}^{n \times n}$ is a symmetric positive semidefinite matrix, the Jacobian matrix $J \in \mathcal{R}^{m \times n}$ has full row rank ($m \leq n$) and y and z are vectors of Lagrange multipliers. Here inequalities are componentwise. The special case $H = 0$ corresponds to the linear programming problem in standard form. Applying a primal-dual interior-point method involves solving, at each iteration, mildly unsymmetric linear systems of order $(2n + m) \times (2n + m)$ of the form

$$\begin{pmatrix} H & J^T & -I \\ J & 0 & 0 \\ -Z & 0 & -X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} f_3 \\ g_3 \\ h_3 \end{pmatrix}, \tag{1.3}$$

where I is the $n \times n$ identity matrix, for the Newton direction $(\Delta x, \Delta y, \Delta z)$. Note that, at each iteration, only the entries of the diagonal matrices X and Z change.

Given the block structure of the matrix and exploiting the fact that X and Z are diagonal matrices with positive entries, there are a number of ways of solving the Newton system (1.3). In particular, we can solve instead the symmetric KKT system

$$\begin{pmatrix} H & J^T & -Z^{\frac{1}{2}} \\ J & 0 & 0 \\ -Z^{\frac{1}{2}} & 0 & -X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ -Z^{\frac{1}{2}} \Delta z \end{pmatrix} = \begin{pmatrix} f_3 \\ g_3 \\ -Z^{\frac{1}{2}} h_3 \end{pmatrix}. \tag{1.4}$$

Alternatively, using block Gaussian elimination, the problem can be reduced to a typical 2×2 saddle-point system of order $(n + m) \times (n + m)$ of the form

$$\begin{pmatrix} H + X^{-1}Z & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} f_2 \\ g_2 \end{pmatrix}, \tag{1.5}$$

where f_2 and g_2 depend on quantities defined in (1.3). This augmented system formulation can be further reduced to yield the $m \times m$ normal equation system

$$J(H + X^{-1}Z)^{-1} J^T \Delta y = g_1, \tag{1.6}$$

where, again, g_1 depends on quantities defined in (1.3). The matrix associated with (1.6) is symmetric positive definite. Thus, as well as being a smaller problem, it offers the potential advantage for a direct method that a Cholesky factorization can be used, which is much more straightforward to implement efficiently than a sparse indefinite factorization that requires the incorporation of numerical pivoting to ensure stability. The downside is that it is necessary to first invert or factor $H + X^{-1}Z$. For linear programming problems (for which $H = 0$) this is trivial and in this case using the normal equations system is usually the preferable option. For quadratic optimization problems with a nontrivial matrix H, the inversion of $H + X^{-1}Z$ may completely destroy the sparsity and make the solution of (1.6) impractical. There exists an important class of separable quadratic optimization problems in which H is a diagonal matrix and therefore the operation $(H + X^{-1}Z)^{-1}$ produces a diagonal matrix and allows for the reduction to normal equations.

Here we consider the indefinite systems (1.4) and (1.5). A number of well-known and widely-used interior-point solvers, including KNITRO [4] and IPOPT [28], exploit the 2×2 block form (1.5). This has been made possible through the development of efficient direct solvers for indefinite systems; indeed,

a number of modern sparse direct solvers (including the packages MA57 [6], HSL_MA86 [10] and HSL_MA97 [11, 13] from the HSL mathematical software library [15]) have been designed with such systems at least partly in mind. However, the memory demands of a direct method generally increases much more rapidly than the problem size so that, for very large problems, direct methods can be unsuitable and an iterative method may then be needed, usually a Krylov subspace-based method (see, for example, [26]). In practice, Krylov subspace methods generally converge very slowly when applied to saddle-point systems and an appropriate preconditioner is needed to accelerate convergence. Over the last 20 years or so, a vast amount of work has been devoted to the development of effective preconditioners for saddle-point problems, with efforts often focused on those arising from partial differential equations (see, for example, [2, Section 10] and also [3] for a later and more concise overview as well as [5]).

It is difficult to design a general-purpose preconditioner and the successful use of iterative methods to solve the systems of equations that arise in interior-point methods depend on fine-tuning. Nevertheless, with increasing problem sizes, interest in the use of iterative methods has been growing over the last decade (see, for example, [5, 7] and the references therein). Iterative methods are particularly attractive for linear systems when it is sufficient to find only an approximate solution (so that their execution can be truncated after a relatively small number of iterations). Such a solution is inexact. Consequently, the interior point method then uses an inexact Newton method. This is discussed by Gondzio [7].

Very recently, there has been renewed interest in the possibility of using the 3×3 block form (1.4), avoiding reducing the system to the 2×2 block form. Greif, Moulding and Orban [9] present a spectral analysis for the 3×3 block form and indicate that it can be preferable to the 2×2 one in terms of eigenvalues and conditioning. Morini, Simoncini and Tani [19] build on this work and present new sharp spectral bounds that also support this view. They also examine, using both theoretical and experimental analysis, the use of constraint and augmented preconditioners for both formulations, broadly concluding that the 2×2 block form may be favourable because of its smaller dimension [20].

Orban [21] generalises the ICFS incomplete Cholesky factorization code of Lin and Moré [17] to symmetric quasi definite matrices [27] and shows that, for a range of practical problems, his incomplete LDL^T factorization may be successfully used as a preconditioner for either the 2×2 or 3×3 block form. Orban does not, however, make a practical comparison between using an incomplete factorization preconditioner for the two forms. Independently, Scott and Tůma [22] have proposed a memory-efficient signed incomplete Cholesky (IC) factorization approach for KKT systems that computes a factorization of the form LDL^T, where D has entries ± 1; this extends their work on IC factorizations for positive-definite systems [23, 24]. Scott and Tůma report encouraging results for some saddle-point systems taken from the University of Florida Sparse Matrix Collection.

The focus of this study is an experimental comparison of factorizing the 2×2 and 3×3 block forms within an interior-point solver. To assess whether for a direct method it can be advantageous to use the 3×3 block form, in Section 3 we present results for the HSL_MA97 direct solver. Then, in Section 4, we compare the effectiveness of preconditioning the two forms using the signed IC factorization preconditioner of Scott and Tůma run with GMRES.

2 Experimental environment

In our experiments, we use a simple interior-point code provided by Tyrone Rees of STFC Rutherford Appleton Laboratory. This allows us to experiment with different solvers. Rather than taking matrices from pre-selected iterations of the interior-point solver, we have chosen to consider all iterations and to report average and/or total statistics. At each iteration, the (incomplete) factorization is recomputed; it is outside the scope of this study to explore the possibility of updating the initial factorization at subsequent iterations in an attempt to improve efficiency.

The test problems are taken from the widely-used CUTEst collection [8]. We have chosen problems for which n is at least 1000 and for which using the direct solver HSL_MA97 gives convergence in a maximum of 100 interior-point iterations. Problems representative of different subcollections within CUTEst are used.

The problems and some of their characteristics are given in the tables in the Appendix.

The implementation of the GMRES(100) algorithm (with right preconditioning) offered by the HSL routine `MI24` is employed, with a relative stopping tolerance of either 10^{-7} or 10^{-10}. These values were chosen for our tests on the basis of experimentation: small tolerance values result in the solution to each linear system being closer to that of the direct solver but may require a large increase in the number of GMRES iterations while a larger value may mean (at least in some cases) that the linear systems are not solved with sufficient accuracy. This is discussed further in Section 4.2. In addition, for each solve with GMRES we impose a limit of 1000 iterations. We observe that we found the small tolerance was needed for the convergence of the interior-point method and we use the same value at each iteration. However, it is possible to use a larger tolerance for the initial iterations and then to reduce the tolerance towards convergence. We have found that this can be effective in practice at improving efficiency but we have not developed a robust scheme for varying the tolerance and so use a fixed tolerance in our tests.

All experiments are performed on our test machine that has two Intel Xeon E5620 processors with 24 GB of memory. The gfortran compiler with option -O3 is used and all computation is performed in double precision. `HSL_MA97` is run in parallel while timings for the iterative methods are serial times; all times are in seconds. A simple sparse matrix-vector product is used (it is beyond the scope of this study to implement efficient parallel routines for these products but doing so could potentially substantially reduce the total time). Note that, since Z and X are diagonal, the extra cost of a matrix-vector product for the $(3,3)$ form compared to that for the $(2,2)$ form is small.

3 Experiments with a sparse indefinite direct solver

Although our main interest is in the use of incomplete factorization preconditioners for the KKT systems that arise in interior-point methods, we first consider the performance of direct solvers. Direct solvers are not commonly used to solve the 3×3 block form. This is because, based on the sparsity patterns of the matrices and assuming the same ordering algorithm is used for each, the factor for the larger 3×3 block form will contain more entries than that for the 2×2 block form. Thus, if numerical stability considerations do not result in significant modifications to the chosen pivot sequence for either of the block forms, the 3×3 one will require more memory, be more expensive to apply in the subsequent solve phase of the solver and, in general, be more expensive to compute. However, a potential difficulty associated with the augmented form (1.5) is that $X^{-1}Z$ may cause ill-conditioning as X and Z have entries that iterate towards zero. This may make it necessary to alter the pivot sequence that was chosen on the basis of the sparsity pattern (that is, it may be necessary to delay pivots and eliminate them later in the factorization when they satisfy the stability tests). This leads to more fill in the factors and more operations to perform the factorization and subsequent solves. It is of interest whether the modifications required for the 2×2 block form are more significant than for the 3×3 block form and whether, as a result, the latter can offer advantages over the former when used with a direct solver.

Consider also the alternative symmetric 3×3 block form

$$\begin{pmatrix} H & J^T & -I \\ J & 0 & 0 \\ -I & 0 & -Z^{-1}X \end{pmatrix} \begin{pmatrix} \widehat{\Delta x} \\ \widehat{\Delta y} \\ \widehat{\Delta z} \end{pmatrix} = \begin{pmatrix} \widehat{f_3} \\ \widehat{g_3} \\ \widehat{h_3} \end{pmatrix}. \tag{3.7}$$

Again, $Z^{-1}X$ may cause ill-conditioning but since only X and Z change at each iteration, only the $n \times n$ $(3,3)$ block changes. Thus, if this block is ordered last in the pivot sequence, this form offers the potential advantage of avoiding recomputing the whole factorization at each iteration. However, ordering the $(3,3)$ block at the end is likely to be a poor choice since this is a diagonal block but in general it will fill in by the time it is pivoted on. The interest is in whether savings from this simple approach to avoiding recomputing the whole factorization can offset the disadvantages caused by ordering the diagonal block last.

The direct solver we use is HSL_MA97 [11]. This is a multifrontal code that employs OpenMP for parallelism on multicore machines. It is designed for both sparse symmetric positive definite and indefinite linear systems, incorporating threshold partial pivoting for numerical stability in the latter case. HSL_MA97 offers a number of options that can be used to tune the performance for particular problems. These include optionally prescaling the matrix as well as a choice of ordering algorithms. We retain the default settings except we use the MC64 scaling option. MC64 finds a maximum matching of an unsymmetric matrix such that the largest entries are moved on to the diagonal; this leads to an unsymmetric scaling such that the scaled matrix has all ones on the diagonal and the remaining entries are of modulus less than or equal to one. The approach is symmetrized by initially ignoring the symmetry of the matrix and then averaging the relevant row and column scalings from the unsymmetric permutation. Our experience has been that, for tough symmetric indefinite linear systems (such as the KKT systems that arise in interior-point methods), this scaling can substantially reduce the number of delayed pivots [14]. The reported HSL_MA97 times include the time for scaling. Note that in many cases, for the size of problem in our test set, the scaling time dominates the factorization time so that, in practice, it can be worthwhile to run without scaling or with a cheaper scaling algorithm, and only employ MC64 if a lot of delayed pivots are reported. Alternatively, it may be possible to use the same scaling for a number of iterations (see [12] for further discussion).

The results for HSL_MA97 are given in columns 2 to 4 in the tables in the Appendix. We report the number of iterations and total time for the interior-point method, the total time for performing the factorization at each iteration using HSL_MA97, the average number of entries in the factor and the average number of delayed pivots.

In column 4 we give results for using (3.7) and ordering the $(3,3)$ block last (the leading $(n+m)\times(n+m)$ submatrix is ordered using HSL_MA97, which chooses either a nested dissection ordering or an approximate minimum degree ordering on the basis of problem size and sparsity). We see that ordering the $(3,3)$ block last leads to much denser factors and the potential savings from reusing the first part of the factorization (the cost of which is approximately that of factorizing the 2×2 block form, given in column 2) are much too small to offset the higher factorization time. Indeed, the times for this approach and the density of the factors are substantially greater than for ordering the 3×3 block form (1.4) without restricting the $(3,3)$ block to the end (column 3).

Comparing (1.5) and (1.4) (columns 2 and 3, respectively), it is clear that using the 3×3 block form leads, as expected, to more entries in the factors and consequently to greater run times. Looking at the average number of delays (av_delays), we do not find that the 2×2 form consistently results in a larger number of delayed pivots than the 3×3 form. Closer inspection also does not show that the number of delayed pivots increases as the interior-point method converges. Thus, the potential ill-conditioning from the $X^{-1}Z$ term is not causing the direct solver difficulties (see [29] for a study of the stability of augmented system factorizations in interior point methods).

Our findings confirm that, for a direct solver, the 2×2 block form should be used.

4 Experiments using an incomplete factorization preconditioner

In this section, we compare using a signed IC factorization for preconditioning the 2×2 and 3×3 block forms. We first briefly describe our limited memory signed IC factorization that is implemented within the package HSL_MI30 from the HSL mathematical software library.

4.1 A signed IC factorization for KKT systems

HSL_MI30 is designed for KKT matrices of the form

$$K = \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}. \tag{4.8}$$

Here A is $n \times n$ symmetric positive definite, B is rectangular $m \times n$ and of full rank ($m \leq n$), and C is $m \times m$ symmetric positive semi-definite. `HSL_MI30` computes a signed incomplete Cholesky factorization. That is, a factorization of the form LDL^T, where L is lower triangular and D is diagonal with $n - m$ positive entries and m negative entries. The matrix K is optionally reordered, scaled and, if necessary, shifted to avoid breakdown of the factorization so that the LDL^T incomplete factorization of the matrix

$$\bar{K} = SQ^T \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} QS + \begin{pmatrix} \alpha(1)I & 0 \\ & -\alpha(2)I \end{pmatrix} \tag{4.9}$$

is computed, where Q is a permutation matrix, S is a diagonal scaling matrix and $\alpha(1:2)$ are non-negative shifts.

A constraint is imposed on the pivot ordering: a pivot corresponding to a variable i in the $(2,2)$ block $-C$ can only be eliminated once all the variables that corresponding to the entries in column i with row index $j \leq n - m$ have been eliminated (in graph terms, a C-node can only be eliminated once all its A-node neighbours have been eliminated). Thus once a fill-reducing ordering has been computed based on the sparsity pattern of K (with all diagonal entries assumed present), it is modified to satisfy the above constraint before the factorization begins.

The algorithm implemented by `HSL_MI30` is a limited memory Tismentsky-Kaporin approach [16, 25]. The Tismenetsky scheme is based on a matrix decomposition of the form

$$\overline{K} = (L + R)\, D\, (L + R)^T - E, \tag{4.10}$$

where L is lower triangular with unit diagonal entries, R is a strictly lower triangular matrix with small entries that is used to stabilize the factorization process, D is a diagonal matrix, and E has the form

$$E = RDR^T + F + F^T, \tag{4.11}$$

where F is strictly lower triangle. E is not computed explicitly and all terms in F are ignored, while the matrix R is used in the computation of L but is then discarded. The user-controlled parameters `lsize` and `rsize` determine the amount of memory used as well as the amount of work involved in computing the incomplete factorization. `lsize` controls the number of entries in the computed incomplete factor L (at most `lsize` fill entries are permitted in each column of L) and `rsize` limits the number of entries in each column of the matrix R. Thus for the 3×3 form the number of entries in L is at most

$$\texttt{lsize} \times (2n + m) + nz(3 \times 3),$$

where $nz(3 \times 3)$ is the number of entries in the lower triangle of (1.4), and the number of entries in R is at most

$$\texttt{rsize} \times (2n + m).$$

Likewise, for the 2×2 form the number of entries in L is at most

$$\texttt{lsize} \times (n + m) + nz(2 \times 2),$$

where $nz(2 \times 2)$ is the number of entries in the lower triangle of (1.5), and the number of entries in R is at most

$$\texttt{rsize} \times (n + m).$$

It is generally advantageous (in terms of the quality of the preconditioner) to use `rsize`>0. Increasing `lsize` and/or `rsize` increases the cost of computing the incomplete factorization (in terms of time and memory). Furthermore, increasing `lsize` leads to a denser L (but one that is often a better preconditioner), increasing the cost of each application of the preconditioner. Setting `lsize` and `rsize` equal to 10 is often a reasonable choice and these are the values we use in our experiments.

Dropping parameters `tau1` and `tau2` may be used to further sparsify L and R, respectively. As each column of L is computed, entries of absolute value less than `tau1` are dropped. These may be included in R but entries less than `tau2` are dropped from R. In our tests, the we set both `tau1` and `tau2` to zero.

In the event of breakdown within the factorization (that is, a pivot is encountered that is smaller in absolute value than a chosen value `small`), a diagonal shift is used [18]. If the breakdown occurs for a pivot in the A-block, a shift $\alpha(1)$ is used; if breakdown occurs in the $(2,2)$ block $-C$, a shift $-\alpha(2)$ is used ($\alpha(1:2) > 0$). It is important to try and use as small a shift as possible but also to limit the number of breakdowns. The user can supply initial shifts $\alpha_0(1:2)$. As we know that there can be zeros on the diagonal of the Hessian matrix H (H is positive semi-definite), we use $\alpha_0(1) = 10^{-8}$ and $\alpha_0(2) = 0$.

Further details of the signed incomplete Cholesky factorization are given in the report [22].

4.2 Numerical findings

Results for the iterative method (`HSL_MI30` used with GMRES(100)) are given in columns 5 to 7 of Tables 1–4 in the Appendix. Here we use an approximate minimum degree ordering (which is post-processed by `HSL_MI30`, as discussed in the previous section), the drop tolerances are set to zero and, for consistency with the direct solver, we use `MC64` scaling. Additionally, we use `lsize`=`rsize`= 10 and the GMRES convergence tolerance (*gmres_tol*) of 10^{-7}. We also tried reducing the amount of fill and reran with `lsize`=`rsize`= 5. We found that the results were the same (or almost the same) for many of the problems and thus in Table 6 we only give results for those problems where reducing `lsize` and `rsize` gave a sparser incomplete factorization and hence different results; for these problems, results are also given for `lsize`=`rsize`= 3 and 1.

Although the alternative form (3.7) was not beneficial for the direct solver, we include it in Tables 1–4 since, again, there is the potential to save work by reusing the incomplete factorization of all but the $(3,3)$ block. The rationale is that, in the incomplete case, the use of a sparsity-preserving ordering is generally less crucial than in the case of a complete factorization. However, we again see that using (3.7) is less effective than using (1.4): the more aggressive dropping that is required results in a poorer quality preconditioner so that, for each solve, a larger number of GMRES iterations are needed and, in a significant number of cases, either GMRES fails to converge (error -2) or the interior-point method does not converge (error -1). We do not consider this form further.

Comparing the 2×2 and 3×3 block forms in Tables 1–4, there is no consistent winner. The 3×3 block form leads, inevitably, to factors that generally have more entries (inevitable since for both forms we allow up to `lsize` fill entries in each column of the factor and the 3×3 block form has more columns). This results in the preconditioner being more expensive both to compute and to apply. For some problems (including AUG3D, AUG3DC and MOSARQP1), the 3×3 form gives a higher quality preconditioner but for others (such as DTOC3 and MARINE) the converse is true. We observe that in some cases (for instance, OPTCDEG3 and STCQP2), using the 2×2 form with `HSL_MI30` leads to the interior-point method requiring a greater number of iterations for convergence than for the 3×3 form, although the total number of GMRES iterations (total_giters) can be less for the former than for the latter. Closer examination shows that, for the 2×2 form, convergence of the interior-point method is slow close to convergence but at each iteration only one step of GMRES is needed. To reduce the total time, we could reuse the preconditioner for a number of interior point iterations or, following [1], we could try and update the preconditioner but this is not something we have explored within our simple interior point code.

For some problems, including MOSARQP1 and MOSARQP2, the interior-point method fails using the 2×2 block form, while with the 3×3 block form it is successful. For these problems and those for which the 2×2 block form requires significantly more interior-point iterations than the 3×3 block form, we rerun with the GMRES convergence tolerance reduced to 10^{-10}. The results are in Table 5. We see that the outer iteration count reduces significantly for the 2×2 block form (although for problem MOSARQP1 the tolerance had to be further reduced to 10^{-12} to reduce the outer iteration count to be competitive to that of the direct solver) and the problems that did not converge with the larger tolerance now converge. For problem DTOC6, the 3×3 block form also now converges. Interestingly, for MOSARQP1 and MOSARQP2, the outer iteration count increases for the 3×3 block form when the smaller tolerance is employed, but only to the same (or nearly the same) number as for the direct solver. Since the smaller tolerance clearly improves reliability, we use *gmres_tol* $= 10^{-10}$ in the rest of our experiments.

The results in Table 6 present a somewhat different picture. These are for a sparser incomplete factorization (`lsize`=`rsize`= 5, 3 and 1). For our test examples, the 3×3 block form leads to the interior-point solver requiring fewer iterations, although if both block forms use the same GMRES tolerance, the total time and the total number of GMRES iterations may still be less for the 2×2 formulation (for example, problem STCQP2). As expected, reducing the memory used for the incomplete factorization reduces the quality of the preconditioner so that the total number of GMRES iterations can increase substantially (illustrated for the 3×3 block form by, for example, problems AUG3D and STCQP2). We observe that, if `rsize` is set to 0 (so that no matrix R is used in the computation of L), this can further reduce the quality. For instance, consider the 3×3 block form with `lsize`= 3: for problem STCQP2, the number of GMRES iterations increases from 19766 with `rsize`= 3 to 31741 with `rsize`= 0, while for MOSARQP2, we fail to get convergence with `rsize`= 0.

As already noted, with the same `lsize` for both forms, the 3×3 block form requires more memory than the 2×2 block form (both in the construction of the factors and in the final L). If we choose `lsize` so that both have similar incomplete factor sizes, the performance of the 2×2 block form compares very favourably. For example, for problem STCQP2 with `lsize`=`rsize`= 10 the factor size for the 2×2 block form is competitive with that for the 3×3 block form with `lsize`=`rsize`= 5. However, while the former requires 1997 GMRES iterations and a total run time of 5.523 seconds, the latter uses 9350 GMRES iterations and takes 28.771 seconds.

It is encouraging to note that, although our direct solver is extremely efficient and is run in parallel, the iterative solver (and particularly the iterative solver applied to 2×2 block form) can report the fastest times; a parallel matrix-vector product routine will reduce the iterative solve times further.

5 Concluding remarks

Limited-memory incomplete Cholesky factorizations have been shown in the last few years to provide efficient and robust preconditioners for a range of problems, including for the "tough" saddle-point systems that arise in interior-point methods [21, 22]. In this study, we have used numerical experiments to compare using the 2×2 and 3×3 block formulations of these systems. As expected, for a direct solver, there appears to be no motivation not to use the smaller 2×2 form but for an iterative solver using a limited-memory incomplete Cholesky factorization preconditioner, the picture is perhaps less clear. We have demonstrated that it is not beneficial to use a simple updating strategy involving ordering the $(3, 3)$ block of (3.7) last in the pivot sequence. But there are examples for which the 3×3 form (1.4) leads to a higher quality preconditioner (which is more expensive to compute but only marginally more expensive to apply) than the 2×2 form (1.5) but for others, the converse is true. At present, for a given problem it is not known a priori which form will lead to the better preconditioner, although some of our numerical results (Table 6) suggest that if the number of fill entries per column allowed for the incomplete factorization is very restricted, 3×3 form may lead to a more favourable preconditioner. Moreover, the 3×3 form appears to be able to use a larger GMRES convergence tolerance without generally adversely effecting the performance of the interior-point method. However, if both the 2×2 and 3×3 block forms are restricted to using the same total amount of memory, the 2×2 form can be better.

Acknowledgements

We are grateful to Tyrone Rees for making available his interior-point code and for a number of helpful discussions on this work. Many thanks to both Nick Gould and Chen Greif for commenting on a draft of this manuscript and to the latter in particular for his encouraging interest in the results, insightful remarks and improvements in the typesetting.

References

[1] S. Bellavia, V. De Simone, D. di Serafino, and B. Morini. A preconditioning framework for sequences of diagonally modified linear systems arising in optimization. *SIAM J. on Numerical Analysis*, 50:3280–3302, 2012.

[2] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[3] M. Benzi and A. Wathen. Some preconditioning techniques for saddle point problems. In *Model Order Reduction: Theory, Research Aspects and Applications*, volume 13 of *Mathematics in Industry*, pages 195–211. Springer, 2008.

[4] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization, G. di Pillo and M. Roma, eds*, pages 35–59. Springer-Verlag, 2006.

[5] M. D'Apuzzo, V. De Simone, and D. di Serafino. On mutual impact of numerical linear algebra and large-sacle optimization with focus on interior point methods. *Computational Optimization and Applications*, 45:283–310, 2010.

[6] 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.

[7] J. Gondzio. Convergence analysis of an inexact feasible interior point method for convex quadratic programming. *SIAM J. on Optimization*, 23(3):1510–1527, 2013.

[8] N. I. M. Gould, D. Orban, and PhL. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads. Technical Report RAL-TR-2013-005, Rutherford Appleton Laboratory, 2013.

[9] C. Greif, E. Moulding, and D. Orban. Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. on Optimization*, 24:49–83, 2014.

[10] J. D. Hogg and J. A. Scott. An indefinite sparse direct solver for large problems on multicore machines. Technical Report RAL-TR-2010-011, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2010.

[11] J. D. Hogg and J. A. Scott. HSL_MA97: a bit-compatible multifrontal code for sparse symmetric systems. Technical Report RAL-TR-2011-024, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2011.

[12] J. D. Hogg and J. A. Scott. On the effects of scaling on the performance of Ipopt. Technical Report RAL-P-2012-009, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2012.

[13] J. D. Hogg and J. A. Scott. New sparse direct solvers for multicore architectures. *Algorithms*, 6:702–725, 2013.

[14] J. D. Hogg and J. A. Scott. Pivoting strategies for tough sparse indefinite systems. *ACM Transactions on Mathematical Software*, 40, 2013. Article 4, 19 pages.

[15] HSL. A collection of Fortran codes for large-scale scientific computation, 2013. `http://www.hsl.rl.ac.uk`.

[16] I. E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$ decomposition. *Numerical Linear Algebra with Applications*, 5:483–509, 1998.

[17] C.-J. Lin and J. J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM J. on Scientific Computing*, 21(1):24–45, 1999.

[18] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation*, 34:473–497, 1980.

[19] B. Morini, V. Simoncini, and M. Tani. Unreduced symmetric KKT systems arising from interior point methods. Part I: spectral estimates. Submitted for publication. Available from `www.optimization-online.org/DB_HTML/2014/05/4356.html`, 2014.

[20] B. Morini, V. Simoncini, and M. Tani. Unreduced symmetric KKT systems arising from interior point methods. Part II: preconditioning. Submitted for publication. Available from `www.optimization-online.org/DB_HTML/2014/07/4418.html`, 2014.

[21] D. Orban. Limited-memory LDLT factorization of symmetric quasi-definite matrices. GERAD Technical Report G-2013-87, 2013.

[22] J. A. Scott and M. Tůma. On signed incomplete Cholesky factorization preconditioners for saddle-point systems. Technical Report RAL-TR-2014-P-003, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2013.

[23] J. A. Scott and M. Tůma. HSL_MI28: an efficient and robust limited memory incomplete Cholesky factorization code. *ACM Transactions on Mathematical Software*, 40, 2014. Article 24, 19 pages.

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

[25] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications*, 154–156:331–353, 1991.

[26] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, UK, 2003.

[27] R. J. Vanderbei. Symmetric quasidefinite matrices. *SIAM J. on Optimization*, 5(1):100–113, 1995.

[28] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 2006. 25–57.

[29] S. Wright. Stability of augmented system factorizations in interior-point methods. *SIAM J. on Matrix Analysis and Applications*, 18:191–222, 1997.

A Tables of results

Results are presented for the test problems in alphabetical order. A † indicates that CUTEst has been used to obtain an approximation to a quadratic programming problem. For each problem, n and m are given together with the number of entries $nz(3\times 3)$ and $nz(2\times 2)$ in the 3×3 and 2×2 block forms (1.4) and (1.5), respectively.

In the tables of results, we use the following notation for the column headings:

- `MA97`(2×2) denotes `HSL_MA97` applied to the 2×2 block form.
- `MA97`(3×3) denotes `HSL_MA97` applied to the 3×3 block form.
- `MA97`$(3\times 3)^*$ denotes `HSL_MA97` applied to the 3×3 block form (3.7). with the $(3,3)$ block ordered last.
- `MI30`(2×2) denotes `HSL_MI30` + GMRES(100) applied to the 2×2 block form.
- `MI30`(3×3) denotes `HSL_MI30` + GMRES(100) applied to the 3×3 block form.
- `MI30`$(3\times 3)^*$ denotes `HSL_MI30` + GMRES(100) applied to the 3×3 block form (3.7) with the $(3,3)$ block ordered last.

In addition, we use the following abbreviations for the statistics that are reported:

- outer_its is the number of interior-point iterations. The limit is 100. If this limit is exceeded, -1 is reported. If GMRES(100) does not converge at some iteration, -2 is reported.
- total_time is the total CPU time (in seconds) for the interior point solver.
- factor_time is the CPU time (in seconds) for the factorizations (respectively, incomplete factorizations) that are computed using `HSL_MA97` (respectively, `HSL_MI30`).
- av_fsize is the average factor size (averaged over the number of interior-point iterations). The number of entries in the factor can vary between iterations but the variation is typically a small percentage of the factor size.
- av_delays is the average number of delayed pivots (direct solver only).
- total_giters is the total number of GMRES iterations performed (incomplete factorization only).

If the interior-point solver failed (either because the maximum number of iterations was reached or GMRES failed to converge), the statistics up to the point of failure are reported with a negative sign prepended.

For problems AUG3D, DTOC2, DTOC6, and GOULDQP2, `HSL_MA97` applied to the 3×3 block form (3.7) exceeded the time limit of 60 minutes; 0's are entered in this case.

The incomplete factorization results in Tables 1–4 use `lsize`=`rsize`= 10 and those in Table 6 use `lsize`=`rsize`= 5.

Table 1: Results for the direct solver and the signed incomplete factorization preconditioned GMRES. `lsize`=`rsize`= 10, $gmres\_tol = 10^{-7}$.

	MA97(2×2)	MA97(3×3)	MA97(3×3)*	MI30(2×2)	MI30(3×3)	MI30(3×3)*
AUG2D $n = 20200$ $m = 10000$ $nz(2 \times 2) = 6.020$E+04 $nz(3 \times 3) = 1.002$E+05						
outer_its	19	17	17	19	17	-1
total_time	0.948	1.478	478.289	0.625	1.532	-36.640
factor_time	0.788	1.299	473.479	0.481	1.334	-20.144
av_fsize	5.138E+05	6.699E+05	2.246E+08	2.702E+05	3.561E+05	-6.206E+05
av_delays	8.858E+01	4.947E+01	2.506E+02	-	-	-
total_giters	-	-	-	41	41	-2850
AUG2DC $n = 20200$ $m = 10000$ $nz(2 \times 2) = 6.020$E+04 $nz(3 \times 3) = 1.006$E+05						
outer_its	15	15	15	15	17	-1
total_time	0.715	1.148	425.773	0.543	1.413	-44.939
factor_time	0.624	1.021	420.957	0.421	1.216	-19.804
av_fsize	5.186E+05	6.729E+05	2.262E+08	2.738E+05	3.561E+05	-6.197E+05
av_delays	0.000E+00	1.067E+00	2.100E+01	-	-	-
total_giters	-	-	-	33	41	-3683
AUG3D $n = 27543$ $m = 8000$ $nz(2 \times 2) = 7.783$E+04 $nz(3 \times 3) = 1.281$E+05						
outer_its	15	16	0	42	16	98
total_time	1.154	2.421	0.000	7.485	5.825	40.624
factor_time	0.847	2.178	0.000	6.579	4.623	36.192
av_fsize	1.495E+06	1.622E+06	0.000E+00	4.517E+05	8.200E+05	7.795E+05
av_delays	-	-	-			
total_giters	-	-	-	373	330	848
AUG3DC $n = 27543$ $m = 8000$ $nz(2 \times 2) = 7.783$E+04 $nz(3 \times 3) = 1.329$E+05						
outer_its	11	10	10	70	10	83
total_time	1.000	1.508	1719.862	14.874	4.140	34.997
factor_time	0.752	1.357	1712.760	13.866	3.410	31.153
av_fsize	1.529E+06	1.679E+06	4.588E+08	4.475E+05	8.489E+05	7.808E+05
av_delays	-	-	-			
total_giters	-	-	-	331	209	719
DTOC1L $n = 5998$ $m = 3996$ $nz(2 \times 2) = 2.598$E+04 $nz(3 \times 3) = 3.797$E+04						
outer_its	6	6	6	6	6	6
total_time	0.053	0.109	563.403	0.038	0.065	1.937
factor_time	0.039	0.070	562.336	0.024	0.039	0.991
av_fsize	9.792E+04	1.713E+05	6.085E+07	5.710E+04	9.441E+04	2.355E+05
av_delays	0.000E+00	0.000E+00	5.033E+05	-	-	-
total_giters	-	-	-	14	22	780
DTOC2† $n = 5998$ $m = 3996$ $nz(2 \times 2) = 3.697$E+04 $nz(3 \times 3) = 4.896$E+04						
outer_its	81	77	0	81	78	-2
total_time	1.264	3.641	0.000	0.888	2.224	-4.417
factor_time	0.919	2.951	0.000	0.680	1.342	-0.371
av_fsize	1.184E+05	1.803E+05	0.000E+00	8.290E+04	1.386E+05	-2.660E+05
av_delays	2.227E+03	4.286E+03	0.000E+00	-	-	-
total_giters	-	-	-	164	850	-2163
DTOC3† $n = 4499$ $m = 2998$ $nz(2 \times 2) = 1.499$E+04 $nz(3 \times 3) = 2.399$E+04						
outer_its	14	11	11	14	11	-2
total_time	0.068	0.118	262.340	0.054	0.075	-1.481
factor_time	0.052	0.094	261.498	0.029	0.039	-0.125
av_fsize	5.580E+04	9.417E+04	3.059E+07	2.730E+04	4.906E+04	-0.000E+00
av_delays	4.006E+02	1.142E+03	1.534E+05	-	-	-
total_giters	-	-	-	38	45	-1000
DTOC6† $n = 10001$ $m = 5000$ $nz(2 \times 2) = 3.000$E+04 $nz(3 \times 3) = 5.000$E+04						
outer_its	35	33	0	-1	-1	-2
total_time	0.717	0.835	0.000	-0.773	-2.343	-9.526
factor_time	0.469	0.503	0.000	-0.402	-0.586	-1.081
av_fsize	1.027E+05	1.674E+05	0.000E+00	-4.040E+04	-8.080E+04	-3.466E+05
av_delays	1.125E+03	0.000E+00	0.000E+00	-	-	-
total_giters	-	-	-	-411	-1192	-2454

Table 2: Results for the direct solver and the signed incomplete factorization preconditioned GMRES (continued). `lsize`=`rsize`= 10, $gmres\_tol = 10^{-7}$.

	MA97(2×2)	MA97(3×3)	MA97(3×3)*	MI30(2×2)	MI30(3×3)	MI30(3×3)*
GASOIL† $n = 10403$ $m = 10398$ $nz(2 \times 2) = 5.879\text{E+}04$ $nz(3 \times 3) = 7.260\text{E+}04$						
outer_its	11	11	11	28	-2	-2
total_time	0.309	0.386	341.996	0.413	-4.880	-6.361
factor_time	0.231	0.310	340.610	0.220	-0.085	-0.864
av_fsize	2.807E+05	3.314E+05	7.529E+07	1.153E+05	-2.708E+05	-0.000E+00
av_delays	4.685E+03	5.339E+03	6.117E+03	-	-	-
total_giters	-	-	-	149	-1122	-1000
GOULDQP2 $n = 19999$ $m = 9999$ $nz(2 \times 2) = 5.999\text{E+}04$ $nz(3 \times 3) = 8.999\text{E+}04$						
outer_its	5	5	0	5	5	-2
total_time	0.209	0.251	0.000	0.089	0.209	-9.511
factor_time	0.123	0.146	0.000	0.051	0.095	-0.388
av_fsize	2.340E+05	3.894E+05	0.000E+00	1.080E+05	2.040E+05	-0.000E+00
av_delays	0.000E+00	1.158E+03	0.000E+00	-	-	-
total_giters	-	-	-	12	30	-1000
HAGER1 $n = 5001$ $m = 2500$ $nz(2 \times 2) = 1.250\text{E+}04$ $nz(3 \times 3) = 2.000\text{E+}04$						
outer_its	9	9	9	9	9	-2
total_time	0.043	0.146	1071.880	0.035	0.071	-1.701
factor_time	0.034	0.128	1070.645	0.019	0.033	-0.178
av_fsize	5.236E+04	9.405E+04	4.981E+07	1.944E+04	3.889E+04	-4.261E+05
av_delays	2.633E+02	3.767E+02	1.002E+06	-	-	-
total_giters	-	-	-	34	55	-1184
HAGER3 $n = 5001$ $m = 2500$ $nz(2 \times 2) = 2.000\text{E+}04$ $nz(3 \times 3) = 3.000\text{E+}04$						
outer_its	25	25	25	25	25	-2
total_time	0.140	0.320	1473.392	0.684	0.355	-1.651
factor_time	0.117	0.277	1471.311	0.619	0.277	-0.104
av_fsize	5.043E+04	9.354E+04	3.652E+07	2.860E+04	5.460E+04	-0.000E+00
av_delays	4.044E+02	9.024E+02	2.627E+05	-	-	-
total_giters	-	-	-	127	78	-1000
HUES-MOD $n = 5000$ $m = 2$ $nz(2 \times 2) = 1.500\text{E+}04$ $nz(3 \times 3) = 2.500\text{E+}04$						
outer_its	17	13	13	17	13	22
total_time	0.092	0.976	17.238	0.056	0.086	13.573
factor_time	0.076	0.958	16.881	0.026	0.058	13.484
av_fsize	1.591E+04	3.772E+04	1.719E+07	1.589E+04	3.770E+04	1.306E+05
av_delays	0.000E+00	0.000E+00	5.346E+02	-	-	-
total_giters	-	-	-	52	29	103
JUNKTURN† $n = 10010$ $m = 7000$ $nz(2 \times 2) = 4.201\text{E+}04$ $nz(3 \times 3) = 6.202\text{E+}04$						
outer_its	4	5	5	4	5	-2
total_time	0.126	0.202	1495.280	0.070	0.178	-5.291
factor_time	0.085	0.137	1493.582	0.043	0.079	-0.406
av_fsize	3.667E+05	5.336E+05	1.625E+08	1.974E+05	2.933E+05	-0.000E+00
av_delays	1.375E+04	1.657E+04	4.808E+03	-	-	-
total_giters	-	-	-	15	54	-1000
LISWET10 $n = 2002$ $m = 2000$ $nz(2 \times 2) = 8.002\text{E+}03$ $nz(3 \times 3) = 1.201\text{E+}04$						
outer_its	8	9	9	8	9	-2
total_time	0.028	0.038	28.225	0.017	0.028	-1.168
factor_time	0.021	0.029	28.067	0.010	0.014	-0.054
av_fsize	3.788E+04	4.565E+04	6.725E+06	1.575E+04	2.667E+04	-1.850E+05
av_delays	1.080E+03	7.047E+02	1.007E+03	-	-	-
total_giters	-	-	-	18	24	-1795
LISWET1 $n = 2002$ $m = 2000$ $nz(2 \times 2) = 8.002\text{E+}03$ $nz(3 \times 3) = 1.201\text{E+}04$						
outer_its	8	8	8	8	8	-2
total_time	0.028	0.033	24.600	0.022	0.025	-0.679
factor_time	0.024	0.025	24.461	0.012	0.015	-0.025
av_fsize	3.790E+04	4.618E+04	6.809E+06	1.575E+04	2.700E+04	-2.220E+05
av_delays	1.083E+03	7.080E+02	1.035E+03	-	-	-
total_giters	-	-	-	18	18	-1038

Table 3: Results for the direct solver and the signed incomplete factorization preconditioned GMRES (continued). `lsize`=`rsize`= 10, $gmres\_tol = 10^{-7}$.

	`MA97`(2×2)	`MA97`(3×3)	`MA97`$(3\times 3)^*$	`MI30`(2×2)	`MI30`(3×3)	`MI30`$(3\times 3)^*$
MARINE† $n = 11215$ $m = 11192$ $nz(2\times 2) = 4.921$E+04 $nz(3\times 3) = 6.382$E+04						
outer_its	21	21	18	71	58	-2
total_time	0.382	0.715	18.338	0.829	159.614	-6.462
factor_time	0.279	0.519	17.744	0.465	0.704	-0.823
av_fsize	2.375E+05	2.890E+05	2.052E+07	9.466E+04	1.564E+05	-0.000E+00
av_delays	2.894E+03	4.800E+03	5.051E+03	-	-	-
total_giters	-	-	-	190	31626	-1000
MOSARQP1 $n = 2500$ $m = 700$ $nz(2\times 2) = 5.967$E+03 $nz(3\times 3) = 1.097$E+04						
outer_its	48	54	43	-1	41	-2
total_time	0.178	0.275	2.502	-0.322	0.248	-0.732
factor_time	0.154	0.234	2.427	-0.238	0.116	-0.088
av_fsize	5.961E+04	7.495E+04	6.090E+05	-2.748E+04	3.339E+04	-1.535E+05
av_delays	1.047E+03	9.851E+02	1.776E+03	-	-	-
total_giters	-	-	-	-223	510	-1111
MOSARQP2 $n = 2500$ $m = 700$ $nz(2\times 2) = 5.967$E+03 $nz(3\times 3) = 1.097$E+04						
outer_its	30	27	28	-1	23	-2
total_time	0.112	0.124	1.760	-0.308	0.117	-0.702
factor_time	0.095	0.103	1.705	-0.205	0.062	-0.083
av_fsize	6.095E+04	7.027E+04	6.161E+05	-2.748E+04	3.401E+04	-1.535E+05
av_delays	1.083E+03	7.533E+02	1.762E+03	-	-	-
total_giters	-	-	-	-219	215	-1118
OPTCDEG3† $n = 4502$ $m = 3000$ $nz(2\times 2) = 1.500$E+04 $nz(3\times 3) = 2.251$E+04						
outer_its	40	36	36	40	36	-2
total_time	0.227	0.239	1531.417	0.173	0.224	-1.525
factor_time	0.190	0.194	1528.663	0.108	0.131	-0.160
av_fsize	5.096E+04	8.418E+04	3.708E+07	2.614E+04	4.625E+04	-0.000E+00
av_delays	1.108E+02	5.019E+02	5.773E+05	-	-	-
total_giters	-	-	-	82	120	-1000
OPTCTRL3† $n = 4502$ $m = 3000$ $nz(2\times 2) = 1.500$E+04 $nz(3\times 3) = 2.400$E+04						
outer_its	15	15	15	15	19	17
total_time	0.098	0.170	177.809	0.072	0.490	1.714
factor_time	0.082	0.120	176.931	0.045	0.070	0.454
av_fsize	6.538E+04	1.067E+05	2.712E+07	2.720E+04	4.737E+04	1.414E+05
av_delays	1.489E+03	2.940E+03	1.531E+03	-	-	-
total_giters	-	-	-	34	749	1040
PINENE† $n = 8805$ $m = 8795$ $nz(2\times 2) = 4.102$E+04 $nz(3\times 3) = 5.179$E+04						
outer_its	16	12	12	17	-2	-2
total_time	0.339	0.299	33.311	0.207	-4.058	-7.551
factor_time	0.253	0.228	32.832	0.117	-0.099	-0.748
av_fsize	1.764E+05	2.187E+05	2.245E+07	7.908E+04	-1.601E+05	-9.936E+05
av_delays	2.249E+03	2.617E+03	2.291E+03	-	-	-
total_giters	-	-	-	85	-1103	-1574
POWELL20 $n = 5000$ $m = 5000$ $nz(2\times 2) = 1.500$E+04 $nz(3\times 3) = 2.500$E+04						
outer_its	13	14	14	19	-2	14
total_time	1.538	0.113	570.179	0.334	-1.571	0.878
factor_time	1.523	0.087	568.991	0.252	-0.052	0.385
av_fsize	6.998E+04	9.642E+04	4.027E+07	3.158E+04	-5.625E+04	1.928E+05
av_delays	-	-	-			
total_giters	-	-	-	163	-1052	497

Table 4: Results for the direct solver and the signed incomplete factorization preconditioned GMRES (continued). `lsize`=`rsize`= 10, $gmres\_tol = 10^{-7}$.

	`MA97`(2×2)	`MA97`(3×3)	`MA97`$(3\times 3)^*$	`MI30`(2×2)	`MI30`(3×3)	`MI30`$(3\times 3)^*$
SOSQP1 $n = 5000$ $m = 2501$ $nz(2\times 2) = 1.750$E+04 $nz(3\times 3) = 2.750$E+04						
outer_its	7	6	6	7	6	6
total_time	0.039	0.044	19.328	0.086	0.049	3.191
factor_time	0.030	0.032	19.134	0.046	0.024	3.148
av_fsize	2.575E+04	5.836E+04	1.989E+07	2.572E+04	5.250E+04	1.808E+05
av_delays	0.000E+00	0.000E+00	8.323E+02	-	-	-
total_giters	-	-	-	131	40	43
STCQP2 $n = 8193$ $m = 4095$ $nz(2\times 2) = 8.600$E+04 $nz(3\times 3) = 1.024$E+05						
outer_its	26	23	23	26	23	39
total_time	1.583	1.483	87.059	5.523	7.657	11.696
factor_time	1.450	1.345	86.439	3.595	4.431	7.320
av_fsize	1.184E+06	1.400E+06	8.831E+06	2.211E+05	3.248E+05	2.943E+05
av_delays	3.123E+03	3.872E+03	1.436E+04	-	-	-
total_giters	-	-	-	1997	2104	2725
ZIGZAG† $n = 3004$ $m = 2500$ $nz(2\times 2) = 1.000$E+04 $nz(3\times 3) = 1.351$E+04						
outer_its	47	62	67	51	64	-2
total_time	0.237	0.410	636.811	0.253	0.560	-1.056
factor_time	0.209	0.346	634.498	0.124	0.251	-0.133
av_fsize	5.958E+04	7.169E+04	1.523E+07	2.470E+04	3.756E+04	-0.000E+00
av_delays	2.811E+02	5.659E+02	1.290E+05	-	-	-
total_giters	-	-	-	548	944	-1000

Table 5: Results for the signed incomplete factorization preconditioned GMRES with `lsize`=`rsize`=10, $gmres\_tol = 10^{-10}$. * denotes run with $gmres\_tol = 10^{-12}$.

	MI3O(2×2)	MI3O(3×3)
AUG3D		
outer_its	14	16
total_time	2.762	6.581
factor_time	1.918	4.754
av_fsize	4.727E+05	8.200E+05
total_giters	413	495
AUG3DC		
outer_its	25	10
total_time	5.564	4.610
factor_time	4.769	3.533
av_fsize	4.589E+05	8.489E+05
total_giters	351	301
DTOC6		
outer_its	35	35
total_time	0.414	2.718
factor_time	0.283	0.209
av_fsize	4.114E+04	8.229E+04
total_giters	122	1496
MOSARQP1		
outer_its	89	49
total_time	0.335	0.323
factor_time	0.225	0.148
av_fsize	2.751E+04	3.326E+04
total_giters	408	736
MOSARQP1*		
outer_its	47	53
total_time	0.206	0.368
factor_time	0.116	0.166
av_fsize	2.778E+04	3.321E+04
total_giters	465	872
MOSARQP2		
outer_its	29	27
total_time	0.115	0.158
factor_time	0.073	0.082
av_fsize	2.814E+04	3.380E+04
total_giters	174	320

Table 6: Results for the signed incomplete factorization preconditioned GMRES with `lsize`=`rsize`=5, `lsize`=`rsize`=3 and `lsize`=`rsize`=1, $gmres\_tol = 10^{-10}$.

	`lsize`=`rsize`=5		`lsize`=`rsize`=3		`lsize`=`rsize`=1	
	`MI30`(2×2)	`MI30`(3×3)	`MI30`(2×2)	`MI30`(3×3)	`MI30`(2×2)	`MI30`(3×3)
AUG2D						
outer_its	-1	17	-1	17	-1	17
total_time	-6.368	1.846	-7.483	5.278	-23.749	103.203
factor_time	-4.781	1.509	-4.470	1.403	-3.720	1.202
av_fsize	-2.234E+05	3.561E+05	-1.624E+05	2.772E+05	-1.014E+05	1.705E+05
total_giters	-599	100	-1864	1502	-5788	12837
AUG2DC						
outer_its	-1	15	-1	15	-1	15
total_time	-7.542	1.412	-8.439	4.240	-22.631	88.861
factor_time	-6.244	1.189	-5.914	1.102	-5.002	0.958
av_fsize	-2.234E+05	3.587E+05	-1.624E+05	2.792E+05	-1.014E+05	1.717E+05
total_giters	-522	62	-1442	1224	-4684	11666
AUG3D						
outer_its	27	16	14	16	76	16
total_time	3.220	5.116	1.965	5.370	6.618	16.090
factor_time	1.967	2.403	0.531	1.679	3.762	1.187
av_fsize	2.665E+05	4.849E+05	2.062E+05	3.508E+05	1.230E+05	2.168E+05
total_giters	711	884	875	1212	1632	3203
AUG3DC						
outer_its	13	10	34	10	-1	10
total_time	2.098	3.742	4.792	3.830	-12.782	11.369
factor_time	1.167	2.110	3.488	1.602	-10.408	1.293
av_fsize	2.838E+05	5.020E+05	1.981E+05	3.632E+05	-1.226E+05	2.244E+05
total_giters	513	526	725	710	-1176	1937
MOSARQP1						
outer_its	-1	49	-2	33	-2	-2
total_time	-0.672	0.323	-1.160	1.817	-2.419	-4.549
factor_time	-0.248	0.147	-0.084	0.092	-0.037	-0.020
av_fsize	-2.287E+04	3.326E+04	-1.870E+04	2.963E+04	-1.172E+04	-2.315E+04
total_giters	-2729	736	-4188	5365	-8673	-9245
MOSARQP2						
outer_its	-1	27	-2	25	-2	-2
total_time	-0.834	0.157	-1.735	1.303	-3.785	-5.015
factor_time	-0.250	0.075	-0.113	0.069	-0.045	-0.025
av_fsize	-2.287E+04	3.380E+04	-1.800E+04	2.991E+04	-1.121E+04	-2.109E+04
total_giters	-3784	320	-6263	3914	-13602	-10355
STCQP2						
outer_its	26	23	26	23	26	23
total_time	14.073	28.771	20.386	56.422	21.794	79.697
factor_time	2.370	2.460	1.719	1.848	0.783	0.811
av_fsize	1.573E+05	2.179E+05	1.318E+05	1.752E+05	1.063E+05	1.325E+05
total_giters	8459	9350	13754	19766	15622	28055