Solving large linear least squares problems with linear equality constraints

BIT Numerical Mathematics
https://doi.org/10.1007/s10543-022-00930-2

BIT

Solving large linear least squares problems with linear equality constraints

Jennifer Scott[1,2] · **Miroslav Tůma**[3]

Received: 24 June 2021 / Accepted: 14 June 2022

Abstract

We consider the problem of solving large-scale linear least squares problems that have one or more linear constraints that must be satisfied exactly. While some classical approaches are theoretically well founded, they can face difficulties when the matrix of constraints contains dense rows or if an algorithmic transformation used in the solution process results in a modified problem that is much denser than the original one. We propose modifications with an emphasis on requiring that the constraints be satisfied with a small residual. We examine combining the null-space method with our recently developed algorithm for computing a null-space basis matrix for a "wide" matrix. We further show that a direct elimination approach enhanced by careful pivoting can be effective in transforming the problem to an unconstrained sparse-dense least squares problem that can be solved with existing direct or iterative methods. We also present a number of solution variants that employ an augmented system formulation, which can be attractive for solving a sequence of related problems. Numerical experiments on problems coming from practical applications are used throughout to demonstrate the effectiveness of the different approaches.

Communicated by Michiel E. Hochstenbach.

J. Scott was partially supported by the EPSRC Grant EP/W009676/1. M. Tůma was supported by project GACR-12719S of the Grant Agency of the Czech Republic.

✉ Jennifer Scott
jennifer.scott@stfc.ac.uk

Miroslav Tůma
mirektuma@karlin.mff.cuni.cz

1 STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, Oxfordshire OX11 0QX, UK

2 School of Mathematical, Physical and Computational Sciences, University of Reading, Reading RG6 6AQ, UK

3 Department of Numerical Mathematics, Faculty of Mathematics and Physics, Charles University, Sokolovska, 49/83, 186 75 Praha 8, Czech Republic

Published online: 05 July 2022

Keywords Sparse matrices · Linear least squares problems · Linear equality constraints · Null space method

Mathematics Subject Classification 65F05 · 65F08 · 65F20 · 65F50

1 Introduction

Our interest lies in efficient and robust methods for solving large-scale linear least squares problems with linear equality constraints. We assume that $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{p \times n}$, with $m > n \gg p$. We further assume that A is large and sparse and C represents a few, possibly dense, linear constraints. Given $b \in \mathbb{R}^m$ and $d \in \mathbb{R}^p$, the least squares problem with equality constraints (the LSE problem) is

$$\min_{x \in \mathbb{R}^n} \|A\,x - b\|_2^2 \tag{1.1}$$

$$\text{s.t. } C\,x = d. \tag{1.2}$$

A solution exists if and only if (1.2) is consistent. For simplicity, we assume that C has full row rank (although the proposed approaches can be made more general). In this case, (1.2) is consistent for any d. A solution to the LSE problem (1.1)–(1.2) is unique if and only if $\mathcal{N}(A) \cap \mathcal{N}(C) = \{0\}$, where for any matrix B, $\mathcal{N}(B)$ denotes its null space. This is equivalent to the extended matrix

$$\mathscr{A} = \begin{pmatrix} A \\ C \end{pmatrix} \tag{1.3}$$

having full column rank. In the case of non-uniqueness, there is a unique minimum-norm solution.

LSE problems arise in a variety of practical applications, including scattered data approximation [13], fitting curves to data [16], surface fitting problems [35], real-time signal processing, and control and communication leading to recursive problems [50], as well as nonlinear least squares problems and least squares problems with inequality constraints. For example, in fitting curves to data, equality constraints may arise from the need to interpolate some data or from a requirement for adjacent fitted curves to match with continuity of the curves and possibly of some derivatives. Motivations for LSE problems together with solution strategies are summarized in the research monographs [7, 8, 29].

Classical approaches for solving LSE problems derive an equivalent unconstrained linear least squares (LS) problem of lower dimension. There are two standard ways to perform this reduction: the null-space approach [21, 29] and the method of direct elimination [10], both of which, with suitable implementation, offer good numerical stability. These methods, termed *constraint substitution* methods, consider the constraints (1.2) as the primary data and substitute from them into the LS problem (1.1). The former performs a substitution using a null-space basis of C obtained from a QR factorization, while the latter is based on substituting an expression for selected solution components from the constraints into (1.1). This can be done using a QR factorization of C [7, 10].

If there are a large number of constraints, a pivoted LU factorization might also be an option [31]. Other solution methods, which may be regarded as complementary to the constraint substitution approaches, reverse the direction of the substitution, substituting from the LS problem into the constraints. This involves the use of an augmented system and include a Lagrange multiplier formulation [22], updating procedures that force the constraints to be satisfied a posteriori [6, 7], and a weighting approach [5, 36, 46],

Solving large-scale LS problems is typically much harder than solving systems of linear algebraic equations, in part because key issues such as ill-conditioning or dense structures within an otherwise sparse problem can vary significantly between different problem classes. Consequently, we do not expect that there will be a single method that is optimal for all LSE problems, and having a range of approaches available that target different problems is important. Our main objective is to revisit classical solution strategies and to propose new ideas and modifications that enable large-scale systems to be solved, with an emphasis first on the possibility that the constraints may be dense, and second on requiring that the constraints be tightly satisfied. In Sects. 2 and 3, we consider the null-space method and the direct elimination approach, respectively. We review the methods and show how they can be used for large-scale problems. In Sect. 4, we present complementary solution approaches within an augmented system framework. This allows us to treat the constraints and the least squares part of the problem using a single extended system of equations or via a global updating scheme. Both direct and iterative methods are discussed.

Much of the published literature related to LSE problems lacks numerical results. For instance, Björck [6] remarks "no attempt has yet been made to implement the (general updating LSE) algorithm", and as far as we are aware, attempts remain absent. We assume this is because implementing the algorithms is far from straightforward. While it is not the intention here to offer a full general comparison of the different approaches, throughout our study we use numerical experiments on problems arising from real applications to highlight key features that may make a method attractive (or unsuitable) for particular problems and to illustrate the effectiveness of the different approaches. Our key findings and recommendations are summarized in Sect. 5.

We end this introduction by describing our test environment. The test matrices are taken from the SuiteSparse Matrix Collection [15] and comprise a subset of those used by Gould and Scott in their study of numerical methods for solving large-scale LS problems [20]. If necessary, the matrix is transposed to give an overdetermined system. Basic information on our test set is given in Table 1.

The problems in the top half of the table contain rows that are identified as dense by Algorithm 1 of [42] (with the density parameter set to 0.05). These rows are taken to form the constraint matrix C and all other rows form A. For the other problems, we form A by removing the 20 densest rows of the SuiteSparse matrix; some or all of these rows are used to form C (and the rest are discarded). Table 1 reports data for $p = 5$ and 20 (denoted, for example, by deter_5 and deter_20, respectively). Although the densest rows are not necessarily very dense, we make this choice because it corresponds to the typical situation in which the constraints couple many of the solution components together. For some of our test examples, splitting the supplied matrix into a sparse part and a dense part results in the sparse part A containing a small number of null columns (at most 7 such columns for our test examples). For the purpose of our experiments,

Table 1 Statistics for our test set. m, n and $nnz(\mathscr{A})$ are, respectively, the row and column counts and the number of entries in the matrix $\mathscr{A}$ given by (1.3). dratio is the ratio of the nonzero counts of the densest row to the sparsest row of $\mathscr{A}$. † indicates at least one column was removed to ensure there are no null columns in A

Identifier	m	n	$nnz(\mathscr{A})$	dratio	$\|x\|_2$	$\|r\|_2$
lp_fit2p	13,525	3,000	50,284	3,000	1.69×10^{1}	1.10×10^{2}
sc205-2r†	62,423	35,212	123,237	1,602	8.76×10^{1}	2.04×10^{2}
scagr7-2b†	13,847	9,742	35,884	1,792	1.11×10^{2}	6.07×10^{1}
scagr7-2r†	46,679	32,846	120,140	6,048	1.82×10^{2}	1.13×10^{2}
scrs8-2r†	27,691	14,357	58,429	2,051	8.57×10^{1}	1.46×10^{2}
sctap1-2b	33,858	15,390	99,454	771	1.46×10^{2}	1.72×10^{2}
sctap1-2r	63,426	28,830	186,366	1,443	1.65×10^{2}	2.07×10^{2}
south31	36,321	18,425	112,328	17,520	2.75×10^{1}	1.88×10^{2}
testbig	31,223	17,613	61,639	802	6.40×10^{1}	1.44×10^{2}
deter3_20	21,777	7,647	44,547	73	1.59×10^{3}	1.22×10^{2}
deter3_5	21,762	7,647	43,807	73	1.57×10^{3}	1.22×10^{2}
fxm4_6_20	47,185	22,400	265,442	24	5.00×10^{2}	9.60×10^{1}
fxm4_6_5	47,170	22,400	265,141	24	5.33×10^{2}	9.59×10^{1}
gemat1_20	10,595	4,929	47,369	22	3.17×10^{4}	8.59×10^{1}
gemat1_5	10,580	4,929	47,339	28	2.44×10^{4}	8.19×10^{1}
stormg2-8_20	11,322	4,393	28,553	21	2.83×10^{1}	7.97×10^{1}
stormg2-8_5	11,307	4,393	28,273	21	3.97×10^{1}	7.78×10^{1}

we remove the corresponding columns from the extended matrix (1.3) (the data in Table 1 is for the modified problem). In all our tests, we check that the norms of the computed solution x and least squares residual $r = b - A\,x$ are consistent with the values given in Table 1.

In our experiments, we prescale the extended matrix $\mathscr{A}$ given by (1.3) by normalizing each of its columns. That is, we replace $\mathscr{A}$ by $\mathscr{A}\mathscr{D}$, where $\mathscr{D}$ is the diagonal matrix with entries $\mathscr{D}_{ii}$ satisfying $\mathscr{D}_{ii} = 1/\|\mathscr{A}e_i\|_2$ (e_i denotes the i-th unit vector). The entries of $\mathscr{A}\mathscr{D}$ are at most one in absolute value. The vectors b and d are set to be vectors of 1's (so that $\|b\|_2$ and $\|d\|_2$ are $O(1)$).

For the substitution approaches described in Sects. 2 and 3, we have developed prototype Fortran codes; in Sect. 4, the augmented system methods are implemented using the SuiteSparseQR package of Davis [14] and Fortran software from the HSL mathematical software library [26]. The prototype codes are not optimised for efficiency and so computational times are not reported. Developing library quality implementations is far from trivial and is outside the scope of the current study, which focuses rather on determining which approaches are sufficiently promising for sophisticated implementations to be considered in the future.

Notation All norms are 2-norms and in the rest of the paper, to simplify the notation, $\|.\|_2$ is denoted by $\|.\|$. I is used to denote the identity matrix of appropriate dimension. The entries of any matrix B are $(B)_{i,j}$ and its columns are denoted by $b_1, b_2, \ldots$. The null space of B is $\mathcal{N}(B)$ and Z is used to denote a matrix whose columns form a basis for the null space (i.e., Z satisfies $B\,Z = 0$). Permutation matrices are denoted by P (possibly with a subscript). The *normal matrix* for (1.1) is $H = A^T\,A$.

2 The null-space approach

The null-space approach is a standard technique for solving least squares problems. It is based on constructing a matrix $Z \in \mathbb{R}^{n\times(n-p)}$ such that its columns form a basis for $\mathcal{N}(C)$. Any $x \in \mathbb{R}^n$ satisfying the constraints can be written in the form

$$x = x_1 + Z\,x_2, \tag{2.1}$$

where $x_1 \in \mathbb{R}^n$ is a particular solution of the underdetermined system $C\,x_1 = d$. The minimum-norm solution can be obtained from the QR factorization of C, that is, $C\,P_C = Q_C\,\begin{pmatrix} R_C & 0\end{pmatrix}$, where the permutation $P_C \in \mathbb{R}^{n\times n}$ represents the pivoting, $R_C \in \mathbb{R}^{p\times p}$ is an upper triangular matrix and $Q_C \in \mathbb{R}^{p\times p}$ is an orthogonal matrix. x_1 is then given by

$$x_1 = P_C \begin{pmatrix} R_C^{-1} Q_C^T\, d \\ 0 \end{pmatrix}.$$

Substituting (2.1) into (1.1) gives the transformed LS problem

$$\min_{x_2} \|A\,Z\,x_2 - (b - A\,x_1)\|^2\,. \tag{2.2}$$

The method is summarized as Algorithm 1.

Algorithm 1 Null-space method for solving the LSE problem (1.1)-(1.2) with C of full row rank.

1: Find $x_1 \in \mathbb{R}^n$ such that $C\,x_1 = d$.
2: Construct $Z \in \mathbb{R}^{n\times(n-p)}$ of full column rank such that $C\,Z = 0$.
3: Solve the normal equations $Z^T H\,Z\,x_2 = (A\,Z)^T(b - A\,x_1)$ corresponding to (2.2)
▷ Here $H = A^T\,A$.
4: Set $x = x_1 + Z\,x_2$.

In the 1970s, the null-space method was developed and discussed by a number of authors, including in relation to quadratic programming [21, 29, 39, 45]. These and subsequent contributions formulate the approach via the orthogonal null-space basis

obtained, for example, from the QR factorization of C^T given by

$$C^T = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix. Z is equal to the last $n - p$ columns of Q and consequently is dense. Note that although it is possible to store Q implicitly using, for example, Householder transformations, the memory demands and implied operation counts are generally too high. Our interest is in large LS problems and therefore it may not be practical to solve the $(n - p) \times (n - p)$ system in Step 3 if Z is dense. To make the approach feasible for large problems we can exploit our recent work [44] on constructing sparse null-space bases of "wide" matrices such as C that have many more columns than rows and may include some dense rows.

Scott and Tůma [44] propose a number of ways to construct sparse Z. In our experiments, we employ Algorithm 3 from Section 3 of [44]. This algorithm first computes a QR factorization of C with column pivoting. The chosen pivots correspond to p columns of R. Then each of the remaining $n - p$ columns of C induces a column $z \in Z$ that is computed independently of the other columns as follows. While in the trivial case of a zero column the corresponding z contains a single nonzero entry, for any nonzero column $c \in C$ a linearly dependent set involving other columns of C is constructed. The smallest such set is called a circuit; circuits play an important role in the problem of the sparsest null-space basis [12]. The coefficients of the linear combination of c and other columns of C that sum to zero are the row entries of the column $z \in Z$ corresponding to c. The linearly dependent sets are found using a partial pivoted QR factorization of C (with at most p steps) that involves the column c. To obtain Z with a narrow bandwidth so that $Z^T H Z$ is sparse when H is sparse, a pivoting threshold $\theta \in [0, 1]$ is employed in these partial QR factorizations. The role of θ is to balance the locality of the dependent sets (combining columns of C whose indices are close to c) with the stability of a QR factorization with column pivoting (which maximizes the absolute values of the diagonal entries of R). Small values of θ result in Z having a narrow bandwidth.

Our first results are for problems deter3 and gemat1. As discussed in the Introduction, we form the constraint matrix C by taking the $p = 2, 5, 10, 20$ densest rows of $\mathscr{A}$. The sparse block A is the same for each case. In Fig. 1, we plot the number of entries $nnz(Z^T H Z)$ in $Z^T H Z$ and the norm of the constraints residual $\|r_c\| = \|d - C x\|$. As expected, $nnz(Z^T H Z)$ increases with θ, and this increase grows with p. This is illustrated further by the results in Table 2. We see that, independently of the choice of θ, for some problems (including lp_fit2p and sctap1-2r) the constraints are not tightly satisfied. This demonstrates an inherent limitation of the null-space approach of [44] that focuses on constructing the columns of Z so as to keep $Z^T H Z$ sparse but does not result in Z having orthogonal columns.

The matrix $Z^T H Z$ in Step 3 of Algorithm 1 is symmetric positive definite. In the above experiments, we employ the sparse direct solver `HSL_MA87` [23] (combined with an approximate minimum degree ordering). However, for large problems, the memory demands mean it may not be possible to use a direct method; this is illustrated by problem south31 with $\theta = 1$. If a preconditioned iterative solver is used instead,

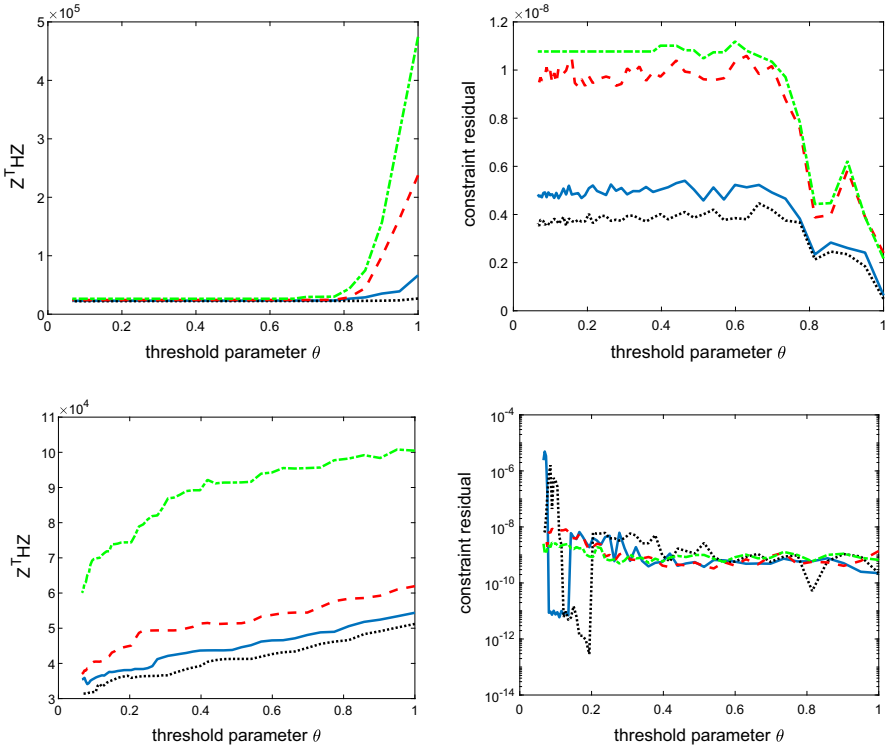

Fig. 1 The number of entries in $Z^T H Z$ (left) and the constraints residual $\|r_c\|$ (right) for problem deter3 (top) and gemat1 (bottom) as the threshold pivoting parameter θ used in the computation of the null-space basis increases from 0.1 to 1. The four curves correspond to $p = 2$ (black dotted line), 5 (blue full line), 10 (red dashed line) and 20 (green dash-dotted line). Observe that using a small θ can improve the sparsity of $Z^T H Z$. For deter3, the constraint residuals are small for all the tested θ and for gemat1, they are satisfactory for $\theta > 0.2$ (colour figure online)

not only are the solver memory requirements much less but explicitly forming the potentially ill-conditioned normal matrix H can be avoided and because Z only needs to be applied implicitly, the need for sparsity can be relaxed. Currently, finding a good preconditioner for use in this case remains an open problem [32].

If a sequence of LSE problems is to be solved with the same set of constraints but different A, the null-space basis can be reused, substantially reducing the work required. But if the constraints are changed, then Z will also change. In [44], we present a strategy that allows Z to be updated when a row (or block of rows) is added to C.

3 The method of direct elimination

The second method we look at is direct elimination [29]. The basic idea is to express the dependency of p selected components of the vector x on the remaining $n - p$

Table 2 The density of $Z^T H Z$ (that is, $nnz(Z^T H Z)/(n-p)^2$) and constraint residual $\|r_c\|$ for two values of the threshold pivoting parameter θ used in the computation of the null-space basis. ‡ indicates insufficient memory for the sparse direct solver `HSL_MA87`

Identifier	p	$\theta = 1$		$\theta = 0.1$	
		Density	$\|r_c\|$	Density	$\|r_c\|$
lp_fit2p	25	0.47	4.14×10^{-5}	0.11	3.07×10^{-5}
sc205-2r	8	0.03	5.25×10^{-8}	0.0002	7.58×10^{-11}
scagr7-2b	7	0.03	1.27×10^{-8}	0.0007	2.79×10^{-8}
scrs8-2c	22	0.31	3.60×10^{-11}	0.23	2.02×10^{-11}
sctap1-2b	34	0.05	3.67×10^{-6}	0.002	4.37×10^{-7}
sctap1-2r	34	0.05	2.76×10^{-3}	0.02	1.83×10^{-4}
south31	5	0.20	‡	0.02	3.26×10^{-7}
testbig	8	0.03	2.53×10^{-11}	0.0002	2.90×10^{-11}
deter3_20	20	0.008	2.58×10^{-9}	0.0004	1.09×10^{-8}
deter3_5	5	0.001	6.39×10^{-10}	0.0004	4.89×10^{-9}
fxm4_6_20	20	0.0006	5.43×10^{-6}	0.0006	6.67×10^{-7}
fxm4_6_5	5	0.0005	7.80×10^{-11}	0.0005	1.10×10^{-11}
gemat1_20	20	0.004	1.10×10^{-9}	0.003	2.40×10^{-9}
gemat1_5	5	0.002	2.24×10^{-10}	0.001	1.00×10^{-11}
stormg2-8_20	20	0.003	7.44×10^{-9}	0.002	7.23×10^{-9}
stormg2-8_5	5	0.002	1.13×10^{-10}	0.002	8.16×10^{-11}

components and to substitute this into the LS problem (1.1). Here we propose how to choose the p components so as to retain sparsity in the transformed problem.

Consider the constraints (1.2). The method starts by permuting and splitting the solution components as follows:

$$C\,x = C\,P_c\,y = \begin{pmatrix} C_1 & C_2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = d,$$

where $P_c \in \mathbb{R}^{n\times n}$ is a permutation matrix chosen so that $C_1 \in \mathbb{R}^{p\times p}$ is nonsingular. Let $A\,P_c = \begin{pmatrix} A_1 & A_2 \end{pmatrix}$ be a conformal partitioning of $A\,P_c$. Substituting the expression

$$y_1 = C_1^{-1}(d - C_2\,y_2) \tag{3.1}$$

into (1.1) gives the transformed LS problem

$$\min_{y_2} \left\| A_T\,y_2 - (b - A_1\,C_1^{-1}d) \right\|^2, \tag{3.2}$$

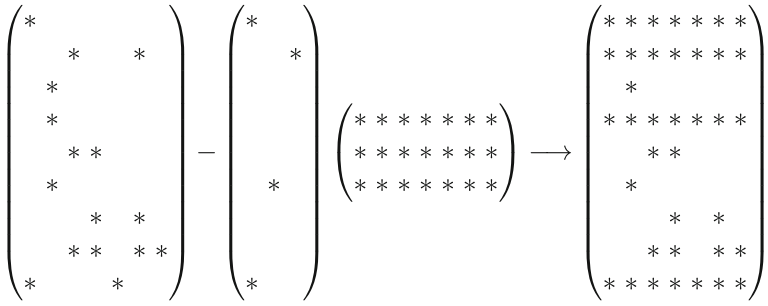

Fig. 2 Example of the transformation in the direct elimination approach. Here $m = 9$, $p = 3$, $n = 7$. The depicted matrices (from the left) represent the transformation $A_T = A_2 - A_1 C_1^{-1} C_2$. The matrix $C_1^{-1} C_2 \in \mathbb{R}^{p \times n}$ is depicted as fully dense

with the transformed matrix

$$A_T = A_2 - A_1 C_1^{-1} C_2 \in \mathbb{R}^{m \times (n-p)}. \tag{3.3}$$

Note that if C_1 is irreducible, the transformation combines all the rows of C_2. If C is composed of dense rows then A_T has more dense rows than A. We thus seek to add as few row patterns as possible replicating the (possibly) dense pattern of C within A_T. If both A and C are sparse, the substitution leads to a sparse LS problem. We have the following straightforward result.

Lemma 3.1 *Let $A \in \mathbb{R}^{m \times n}$ be sparse. Let $m > n > p$ and assume a conformal column splitting induced by the permutation P_c is such that $C P_c = \begin{pmatrix} C_1 & C_2 \end{pmatrix}$ and $A P_c = \begin{pmatrix} A_1 & A_2 \end{pmatrix}$ with $C_1 \in \mathbb{R}^{p \times p}$ nonsingular and $A_1 \in \mathbb{R}^{m \times p}$. Define the index set*

$$Occupied = \{i \mid (A_1)_{i,k} \neq 0 \text{ for some } k,\ 1 \leq k \leq p\}.$$

Then the number of dense rows in A_T given by (3.3) is at most the number of entries in $Occupied$.

Proof The result follows directly from the transformation. Assuming the rows of $C_1^{-1} C_2$ are dense, the substitution step (3.1) of the direct elimination implies a dense row k in A_T only if there is a nonzero in the k-th row of A_1. □

A simple example is given in Fig. 2. Here we ignore cancellation of nonzeros during arithmetic operations. We see that the pattern of A_T satisfies Lemma 3.1. Note that, although in this example $C_1^{-1} C_2$ is shown as dense, it need not be fully dense and the number of entries in $Occupied$ represents an upper bound on the number of dense rows in A_T.

Lemma 3.1 implies that the LSE problem is transformed to a LS problem (3.2) that has some dense rows, which we refer to as a *sparse-dense* LS problem. Consequently, existing methods for sparse-dense LS problems can be used, including those recently

proposed in [40, 41, 43] (see also the recent direct LS solver HSL_MA85 from the HSL library). A straightforward algorithmic implication of the lemma is that the permuting and splitting of C cannot be separated from considering the sparsity pattern of A because the splitting also determines A_1 and A_2. Thus we want to permute the columns of C to allow a sufficiently well-conditioned factorization of C_1 while limiting the number of entries in $Occupied$ and hence the number of dense rows in A_T. The approach outlined in Algorithm 2 is one way to achieve this. There is an important difference between the pivoting used in Algorithm 3 of [44] (which we used in the previous section) and that of Algorithm 2 below. The former modifies the column pivoting that is considered as standard for QR factorizations by employing a threshold parameter θ that ensures Z is banded and the transformed normal matrix $Z^T H Z$ retains sparsity. The choice of θ aims to balance the stability of the factorization with the sparsity of Z. The threshold parameter $\tau \in (0, 1]$ used in Algorithm 2 also guarantees the pivots in the QR factorization of C are sufficiently large but the selection of the candidate pivots is balanced with limiting the fill-in in the transformed matrix A_T. A crucial role is played by the set of rows held in $Occupied$ that potentially cause fill-in in A_T. The use of different notation for the threshold parameters emphasises the difference between the two QR-based approaches and the distinct roles of the two thresholds.

Algorithm 2 Assume $C = (c_1, \ldots, c_n) \in \mathbb{R}^{p\times n}$ $(p < n)$ has full row rank. Compute $C_1 \in \mathbb{R}^{p\times p}$ and the column permutation $P_c \in \mathbb{R}^{n\times n}$ for the direct elimination method for solving the LSE problem (1.1)–(1.2). P_c is determined by a QR factorization with threshold pivoting; $\tau \in (0, 1]$ is the threshold pivoting parameter.

Initialise: $Occupied = \emptyset$, $S = \emptyset$, and $w_j = \|c_j\|^2$, $j = 1, \ldots, n$. Define $E_n = \{1, 2, \ldots, n\}$.
for $l = 1, ..., p$
  Find $j_{max} = \text{argmax}_{j\in E_n}\{w_j \mid j \in E_n \setminus S\}$.
  Define $E_\tau = \{j \in E_n \setminus S \mid w_j \geq \tau w_{jmax}\}$.
  Find $k \in E_n \setminus S$ such that $k = \text{argmin}_{j\in E_\tau} |\{i \mid (A)_{i,j} \neq 0\} \setminus Occupied|$.
  For $j \in E_n \setminus S$ set $c_j \leftarrow c_j - q_k^T c_j\, q_k$, where $q_k = c_k/\|c_k\|$.
  For $j \in E_n \setminus S$ set $w_j \leftarrow w_j - (q_k^T c_j)^2$.
  Update $S \leftarrow S \cup \{k\}$.
  Update $Occupied \leftarrow Occupied \cup \{i \mid (A)_{i,k} \neq 0\}$.
end for
Set P_c to permute the columns of C with indices in S to obtain C_1.

Observe that the pivoting strategy in Algorithm 2 considers C and A simultaneously and will not select a column as the pivot column if this column in A is dense (as it would lead to A_T being dense). While we do not discuss the implementation details, we remark that care is needed to ensure efficiency. For example, the QR factorization with pivoting of a wide matrix is relatively cheap but it may be necessary to store the

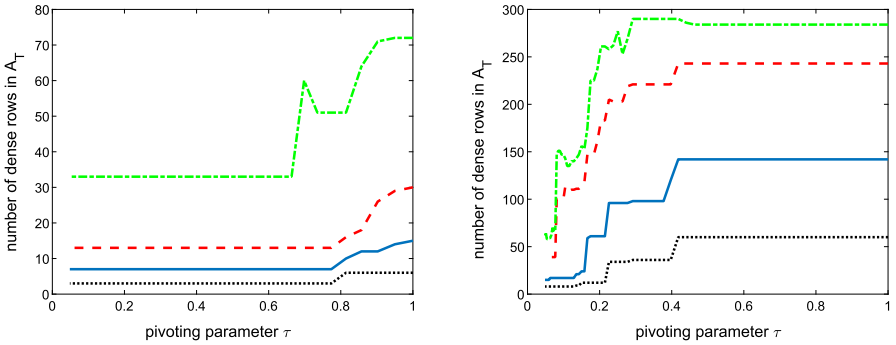

Fig. 3 The number of dense rows in the transformed matrix A_T as the parameter τ increases from 0.05 to 1 for problems deter3 (left) and gemat1 (right). The four curves correspond to $p = 2$ (black dotted line), 5 (blue full line), 10 (red dashed line) and 20 (green dash-dotted line) (colour figure online)

squares of the column norms using a heap, which is why we emphasize their role in the algorithm by using the explicit notation w_i for these norms.

The effects of increasing the pivoting parameter τ on the number of dense rows in A_T are illustrated in Fig. 3 for problems deter3 and gemat1; results for the full test set are given in Table 3. The dense rows of the transformed matrix A_T are determined using Algorithm 1 of [42] and to solve the transformed LS problem (3.2) we use the sparse-dense preconditioned iterative approach of [40].

This computes a Cholesky factorization of the normal matrix corresponding to the sparse part of A_T and uses it as a preconditioner within a conjugate gradient (CG) method; the CG convergence tolerance that measures the relative decrease of the transformed residual $\|A_T^T r\|/\|r\|$ is set to 10^{-11}. For the problems in the top half of the table for which the rows of C are much denser than those of A (recall Table 1), reducing τ leads to only a small reduction in the number ndense of dense rows in A_T. However, when the constraints are not dense (the problems in the lower half of the table), ndense can be significantly decreased by choosing $\tau < 1$, although if τ is too small, the matrix C_1 computed by Algorithm 2 can become highly ill-conditioned and A_T close to being singular. In our experiments we occasionally observed this for $\tau < 10^{-5}$.

By comparing the pairs of problems in the lower half of the table (such as deter3_20 and deter3_5) and considering the plots in Fig. 3, we see that increasing the number p of constraints can lead to a sharp increase in ndense (even if these constraints are relatively sparse), which can result in the transformed problem being hard to solve. The constraints are very well satisfied in all the test cases, making this an attractive approach if a good sparse-dense LS solver is available and the number of dense rows in the transformed problem is not too large. Furthermore, it can be used, without modification, if the matrix A contains a (small) number of dense rows. However, for a sequence of problems, if A and/or C changes then, because direct elimination couples the two matrices, the computation must be completely restarted.

Table 3 The number (ndense) of dense rows in A_T and norm of the constraints residual $\|r_c\|$ for two values of the pivoting parameter τ

Identifier	p	$\tau = 1$		$\tau = 0.1$	
		ndense	$\|r_c\|$	ndense	$\|r_c\|$
lp_fit2p	25	115	8.12×10^{-12}	100	6.77×10^{-11}
sc205-2r	8	9	3.32×10^{-14}	7	6.13×10^{-14}
scagr7-2b	7	14	4.03×10^{-13}	6	1.48×10^{-13}
scrs8-2c	22	16	6.80×10^{-14}	16	1.25×10^{-13}
sctap1-2b	34	72	7.60×10^{-13}	63	1.31×10^{-12}
sctap1-2r	34	66	1.90×10^{-13}	57	1.40×10^{-13}
south31	5	20	2.57×10^{-15}	16	3.45×10^{-14}
testbig	8	9	4.09×10^{-15}	8	1.06×10^{-14}
deter3_20	20	72	8.02×10^{-14}	33	5.75×10^{-14}
deter3_5	5	15	8.02×10^{-14}	7	1.53×10^{-13}
fxm4_6_20	20	113	2.07×10^{-14}	80	5.70×10^{-14}
fxm4_6_5	5	50	7.81×10^{-16}	14	9.13×10^{-16}
gemat1_20	20	284	8.95×10^{-15}	147	1.64×10^{-14}
gemat1_5	5	142	9.77×10^{-14}	17	4.61×10^{-14}
stormg2-8_20	20	136	5.30×10^{-14}	94	2.29×10^{-15}
stormg2-8_5	5	61	3.28×10^{-15}	35	1.50×10^{-14}

4 Approaches described via augmented systems

We now focus on complementary approaches that are based on substitution from the unconstrained least squares problem into the constraints. A useful way to describe this is via the augmented (or saddle-point) system

$$\begin{pmatrix} H & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} A^T b \\ d \end{pmatrix}, \qquad H = A^T A. \tag{4.1}$$

Here $\lambda \in \mathbb{R}^p$ is a vector of additional variables that are often called *Lagrange multipliers* [18, 22]. The solution x of (4.1) solves the LSE problem. Using (4.1) can be particularly useful if C is dense and p is small. As we see in the following discussions, this is because the work involved in the proposed algorithms that depends upon p is effectively independent of the density of C. Observe that because (4.1) has a zero $(2, 2)$ block, the augmented system can be also used to give an alternative derivation of the null-space approach of Algorithm 1. For if Z is such that $C\,Z = 0$ and x_1 is a particular solution of the second equation of (4.1) so that $Cx_1 = d$ (steps 1 and 2 of

Algorithm 1), then if $x = x_1 + \hat{x}$, (4.1) becomes

$$\begin{pmatrix} H & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} \hat{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} A^T(b - A\,x_1) \\ 0 \end{pmatrix}.$$

The second equation in this system is equivalent to finding x_2 such that $\hat{x} = Z\,x_2$. Substituting this into the first equation $H\,Z\,x_2 + C^T\lambda = A^T(b - A\,x_1)$. Hence $Z^T H\,Zx_2 = (A\,Z)^T(b - A\,x_1)$, as in Algorithm 1.

4.1 Direct use of Lagrange multipliers

Algorithm 3 presents a straightforward updating scheme for solving the LSE problem using Lagrange multipliers and (4.1). Any appropriate direct or iterative method can be used for Step 1, which is usually the most expensive part of the computation.

Algorithm 3 Straightforward updating approach based on Lagrange multipliers for solving the LSE problem (1.1)-(1.2) with C having full row rank

1: Solve the sparse unconstrained LS problem $\min_y \|A\,y - b\|^2$
2: Solve $H\,J = C^T$ for $J \in \mathbb{R}^{n\times p}$.
3: Set $Y = C\,J$.
4: Solve $Y\lambda = C\,y - d$ for λ. ▷ Note that $Y \in \mathbb{R}^{p\times p}$ is symmetric positive definite.
5: Set $x = y - J\lambda$.

There is no dependence on C so the solution y does not need to be recomputed when C changes. The method used to solve the system with a block of p right-hand sides in Step 2 can be chosen to exploit Step 1. For example, a sparse Cholesky factorization of H may be computed in Step 1 and the factors reused in Step 2. Using existing sparse LS solvers (and a dense linear solver for the $p \times p$ at Step 4), Algorithm 3 is straightforward to implement and the solution y of the unconstrained LS problem obtained from Step 1 can be compared with that of the LSE computed in Step 5.

As discussed by Golub [17] and Heath [22], a numerically superior direct method that avoids both forming the potentially ill-conditioned normal matrix H and computing the multipliers λ can be derived using a QR factorization of A. Following [43], we obtain Algorithm 4. Here P is a permutation matrix chosen to ensure sparsity of the R factor. Note that, unless b (and hence f) changes, the Q factor need not be retained and the R factor can be reused if the constraints change but A is fixed.

Results for Algorithm 4 presented in Table 4 confirm that the computed solution is such that the norm of the constraints residual $\|r_c\| = \|d - C\,x\|$ is small. We omit results for problems such as deter_5 that have $p = 5$ constraints because they are similar (with $\|r_c\|$ typically smaller than for the corresponding problems with $p = 20$).

Algorithm 4 QR algorithm with updating for solving the LSE problem (1.1)–(1.2) with C having full row rank

1: Compute the QR factorization $\begin{pmatrix} A P & b \end{pmatrix} = Q \begin{pmatrix} R & f \\ 0 & g \end{pmatrix}$ using a sparse QR solver.
2: Solve $R P^T y = f$ for y.
3: Solve $P R^T K^T = C^T$ for $K^T \in \mathbb{R}^{n \times p}$.
4: Compute the minimum-norm solution of $K u = d - C y$. ▷ Use QR factorization of K^T
5: Solve $R P^T z = u$ for z.
6: Set $x = y + z$.

Table 4 Norm of the constraint residuals $\|r_c\|$ for QR with updating (Algorithm 4)

Identifier	$\|r_c\|$	Identifier	$\|r_c\|$	Identifier	$\|r_c\|$
lp_fit2p	4.49×10^{-11}	sctap1-2b	4.42×10^{-11}	deter3_20	1.26×10^{-12}
sc205-2r	4.30×10^{-10}	sctap1-2r	7.62×10^{-11}	fxm4_6_20	8.49×10^{-14}
scagr7-2b	1.36×10^{-11}	south31	4.50×10^{-13}	gemat1_20	2.94×10^{-12}
scagr7-2r	2.18×10^{-11}	testbig	8.43×10^{-11}	stormg2-8_20	2.44×10^{-12}
scrs8-2r	8.63×10^{-11}				

4.2 An extended augmented system approach

An equivalent formulation of (4.1) is given by the 3-block saddle-point system (the first order optimality conditions)

$$\mathscr{A}_{aug}\, y = b_{aug},$$

where

$$\mathscr{A}_{aug} = \begin{pmatrix} I & 0 & A \\ 0 & 0 & C \\ A^T & C^T & 0 \end{pmatrix}, \qquad y = \begin{pmatrix} r \\ -\lambda \\ x \end{pmatrix}, \qquad b_{aug} = \begin{pmatrix} b \\ d \\ 0 \end{pmatrix}. \tag{4.2}$$

Applying the analysis of Section 5 of [43] to this problem yields Algorithm 5. In exact arithmetic, the main difference between the work required by Algorithms 4 and 5 is that the former involves an additional solve with RP^T. For both algorithms, K is independent of b and d.

Algorithm 5 Solve the LS problem (1.1)–(1.2) with C having full row rank using the 3-block augmented system (4.2)

1: Compute the sparse QR factorization $\begin{pmatrix} A P & b \end{pmatrix} = Q \begin{pmatrix} R & f \\ 0 & g \end{pmatrix}$.
2: Solve $P R^T K^T = C^T$ for $K^T \in \mathbb{R}^{n \times p}$.
3: Compute the minimum-norm solution of $K u = d - K f$. ▷ Use QR factorization of K^T
4: Solve $R P^T x = f + u$ for x.

4.3 Augmented regularized normal equations

The next approach weights the constraints and uses a regularization parameter within an augmented system formulation and then aims to balance these two modifications. Consider the weighted least squares problem (WLS)

$$\min_x \left\| A_\gamma x_\gamma - b_\gamma \right\|^2 \quad \text{with} \quad A_\gamma = \begin{pmatrix} A \\ \gamma C \end{pmatrix}, \quad b_\gamma = \begin{pmatrix} b \\ \gamma d \end{pmatrix}, \tag{4.3}$$

for some large γ ($\gamma \gg 1$). Let x_{LSE} be the solution of the LSE problem (1.1)–(1.2). Then because

$$\lim_{\gamma \to \infty} x_\gamma = x_{LSE},$$

the WLS problem can be used to solve the LSE problem approximately [28]. An obvious solution method is to solve the normal equations for (4.3):

$$H_\gamma x = A_\gamma^T A_\gamma x = (A^T A + \gamma^2 C^T C) x = A^T b + \gamma^2 C^T d = A_\gamma^T b_\gamma.$$

The appeal is that no special methods are required: software for solving standard normal equations can be used. However, for very large values of γ, the normal matrix H_γ becomes extremely ill-conditioned; this is discussed in Section 4 of [9], where it is shown that the method of normal equations can break down if $\gamma > \epsilon^{-1/2}$ (ϵ is the machine precision). Furthermore, if C contains dense rows then H_γ will be dense.

Another possibility is to use the regularized normal equations

$$(H_\gamma + \omega^2 I) x = A_\gamma^T b_\gamma, \tag{4.4}$$

where $\omega > 0$ is a regularization parameter [49]. Solving (4.4) is equivalent to solving the $(m + p + n) \times (m + p + n)$ augmented regularized normal equations

$$\mathscr{A}(\omega, \gamma) \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b_\gamma \\ 0 \end{pmatrix}, \qquad \mathscr{A}(\omega, \gamma) = \begin{pmatrix} \omega I & A_\gamma \\ A_\gamma^T & -\omega I \end{pmatrix}, \tag{4.5}$$

where $y = \omega^{-1}(b_\gamma - A_\gamma\, x) \in \mathbb{R}^{m+p}$. The spectral condition number of (4.5) is

$$\mathrm{cond}(\mathscr{A}(\omega,\gamma)) = \sqrt{\mathrm{cond}(H_\gamma + \omega^2\, I)}$$

and Saunders [38] shows that $\mathrm{cond}(\mathscr{A}(\omega,\gamma)) \approx \|A_\gamma\|/\omega$ regardless of the condition of A_γ. Thus using (4.5) potentially gives a significantly more accurate approximation to the pseudo solution $x = A_\gamma^+\, b_\gamma$ (where $(.)^+$ denotes the Moore-Penrose pseudoinverse of a matrix) compared to the approximation provided by solving (4.4). In [48], the parameters are set to $\omega = 10^{-q}$ and $\gamma = 10^q$, where

$$q = \min\{k : 10^{-2k} \le \nu^{-t}\}.$$

Here t-bit floating-point arithmetic with base ν is used.

Rewriting (4.5) using (4.3) and a conformal partitioning of y gives

$$\begin{pmatrix} \omega I & 0 & A \\ 0 & \omega I & \gamma C \\ A^T & \gamma C^T & -\omega I \end{pmatrix} \begin{pmatrix} y_s \\ y_c \\ x \end{pmatrix} = \begin{pmatrix} b \\ \gamma d \\ 0 \end{pmatrix}. \tag{4.6}$$

This system can be solved as in [43] using a modified version of Algorithm 5. Or, eliminating y_s and setting $\omega\gamma = 1$, yields

$$\begin{pmatrix} -H(\omega) & C^T \\ C & \omega^2 I \end{pmatrix} \begin{pmatrix} x \\ y_c \end{pmatrix} = \begin{pmatrix} -A^T\, b \\ d \end{pmatrix}, \qquad H(\omega) = A^T\, A + \omega^2\, I. \tag{4.7}$$

We can solve this system using a QR factorization of $\begin{pmatrix} A \\ \omega I \end{pmatrix}$ and modifying Algorithm 4. Or, ignoring the block structure, we can treat it as a sparse symmetric indefinite linear system and compute an LDL^T factorization (with L unit lower triangular and D block diagonal with blocks of size 1 and 2) using a sparse direct solver such as `HSL_MA97` [24] that incorporates pivoting for stability with a sparsity-preserving ordering. This factorization would have to be recomputed for each new set of constraints. Alternatively, a block signed Cholesky factorization of (4.7) can be used, that is,

$$\begin{pmatrix} -H(\omega) & C^T \\ C & \omega^2 I \end{pmatrix} = \begin{pmatrix} L & \\ B & L_\omega \end{pmatrix} \begin{pmatrix} -I & \\ & I \end{pmatrix} \begin{pmatrix} L^T & B^T \\ & L_\omega^T \end{pmatrix},$$

where

$$H(\omega) = L\, L^T, \quad L\, B^T = -C^T \quad \text{and} \quad S = \omega^2 I + B\, B^T = L_\omega\, L_\omega^T.$$

We then obtain Algorithm 6. Note that B need not be computed explicitly. Rather, the Schur complement S may be computed using $\omega^2 I + C\, L^{-T} L^{-1} C^T$, and $w = B\, z$ may be computed by solving $L\, v = z$ and then setting $w = -C\, v$, and $w = -B^T\, y_c$ may be obtained by solving $Lw = C^T\, y_c$.

Algorithm 6 Given $\omega > 0$, solve the augmented system (4.7) using Cholesky factorizations.

1: Compute the sparse Cholesky factorization $H(\omega) = L\,L^T$.
2: Solve $L\,z = A^T\,b$.
3: Solve $L\,B^T = -C^T$.
4: Form the symmetric positive definite Schur complement $S = \omega^2 I + B\,B^T$ and factorize it $S = L_\omega\,L_\omega^T$.
5: Solve $L_\omega\,v = d + B\,z$ then solve $L_\omega^T\,y = v$.
6: Solve $L^T\,x = z - B^T\,y_c$.

Results for Algorithm 6 for three of our test problems using a range of values of ω are given in Table 5. Note that here $\|r_c\|$ is computed using $r_c = d - C\,x$ (rather than using $r_c = \omega * y_c$). We see that, provided ω is sufficiently small, the values of $\|x\|$ and $\|r\|$ are consistent with those given in Table 1.

By replacing the Cholesky factorization of $H(\omega)$ by an incomplete factorization $H(\omega) \approx \tilde{L}\,\tilde{L}^T$, we can obtain a preconditioner for solving (4.7). In particular, the right-preconditioned system is

$$\begin{pmatrix} -H(\omega) & C^T \\ C & \omega^2 I \end{pmatrix} M^{-1} \begin{pmatrix} w \\ w_c \end{pmatrix} = \begin{pmatrix} -A^T\,b \\ d \end{pmatrix}, \qquad M \begin{pmatrix} x \\ y_c \end{pmatrix} = \begin{pmatrix} w \\ w_c \end{pmatrix}, \tag{4.8}$$

and we can take the preconditioner in factored form to be

$$M = \begin{pmatrix} \tilde{L} & \\ \tilde{B} & I \end{pmatrix} \begin{pmatrix} -I & \\ & \tilde{S}_d \end{pmatrix} \begin{pmatrix} \tilde{L}^T & \tilde{B}^T \\ & I \end{pmatrix}, \tag{4.9}$$

with

$$\tilde{L}\,\tilde{B}^T = -C^T \quad \text{and} \quad \tilde{S} = \omega^2 I + \tilde{B}\,\tilde{B}^T.$$

As the preconditioner (4.9) is indefinite, it needs to be used with a general non-symmetric iterative method such as GMRES [37]. A positive definite preconditioner for use with MINRES [33] can be obtained by replacing $-I$ in (4.9) by I. MINRES has the important advantage of only requiring three vectors of length equal to the size of the linear system. GMRES results are included in Table 5. The GMRES convergence tolerance is taken to be 10^{-11}. We see that the GMRES iteration count is essentially independent of ω. We also ran MINRES with the same settings. For problems sctap1-2r, south31 and deter3_20 with $\omega = 10^{-5}$ the counts were 17, 772 and 56 (approximately twice the GMRES counts). This would be of more interest if all counts were higher.

Our findings in Sect. 4 suggest that, if we require the constraints to be solved with a small residual, then an augmented system based approach combined with a QR factorization performs better (in terms of $\|r_c\|$) than combining it with regularization

Table 5 Results for the augmented regularized normal equations approach (Algorithm 6) for problems sctap1-2r, south31, and deter3_20 using a range of values of ω. iters is the number of preconditioned GMRES iterations. The computed $\|x\|$ and $\|r\|$ are consistent for both approaches

Identifier	ω	$\|x\|$	$\|r\|$	Algorithm 6	GMRES	
				$\|r_c\|$	iters	$\|r_c\|$
sctap1-2r	1.0×10^{-2}	1.44×10^{2}	1.91×10^{2}	5.07×10^{-1}	6	5.07×10^{-1}
	1.0×10^{-3}	1.65×10^{2}	2.07×10^{2}	7.38×10^{-3}	6	7.38×10^{-3}
	1.0×10^{-4}	1.65×10^{2}	2.07×10^{2}	7.42×10^{-5}	6	7.42×10^{-5}
	1.0×10^{-5}	1.65×10^{2}	2.07×10^{2}	7.71×10^{-7}	6	7.42×10^{-7}
	1.0×10^{-6}	1.65×10^{2}	2.07×10^{2}	1.15×10^{-7}	2	7.42×10^{-9}
	1.0×10^{-7}	1.65×10^{2}	2.07×10^{2}	1.08×10^{-7}	6	7.42×10^{-11}
	1.0×10^{-8}	1.65×10^{2}	2.07×10^{2}	1.20×10^{-7}	6	7.64×10^{-13}
	1.0×10^{-9}	1.65×10^{2}	2.07×10^{2}	1.29×10^{-7}	6	4.09×10^{-13}
south31	1.0×10^{-2}	2.75×10^{1}	1.88×10^{2}	8.34×10^{-5}	311	8.34×10^{-5}
	1.0×10^{-3}	2.75×10^{1}	1.88×10^{2}	8.34×10^{-7}	337	7.34×10^{-7}
	1.0×10^{-4}	2.75×10^{1}	1.88×10^{2}	8.34×10^{-9}	352	8.34×10^{-9}
	1.0×10^{-5}	2.75×10^{1}	1.88×10^{2}	8.31×10^{-11}	354	8.85×10^{-11}
	1.0×10^{-6}	2.75×10^{1}	1.88×10^{2}	1.02×10^{-12}	354	1.06×10^{-11}
	1.0×10^{-7}	2.75×10^{1}	1.88×10^{2}	1.84×10^{-13}	354	1.07×10^{-11}
	1.0×10^{-8}	2.75×10^{1}	1.88×10^{2}	1.29×10^{-13}	354	1.07×10^{-11}
	1.0×10^{-9}	2.75×10^{1}	1.88×10^{2}	6.77×10^{-14}	354	1.08×10^{-11}
deter3_20	1.0×10^{-2}	1.22×10^{3}	1.23×10^{2}	6.88×10^{-4}	34	6.88×10^{-4}
	1.0×10^{-3}	1.58×10^{3}	1.22×10^{2}	6.83×10^{-6}	36	6.83×10^{-6}
	1.0×10^{-4}	1.59×10^{3}	1.22×10^{2}	6.83×10^{-8}	36	6.83×10^{-8}
	1.0×10^{-5}	1.59×10^{3}	1.22×10^{2}	6.83×10^{-10}	36	6.83×10^{-10}
	1.0×10^{-6}	1.59×10^{3}	1.22×10^{2}	6.93×10^{-12}	36	6.72×10^{-12}
	1.0×10^{-7}	1.59×10^{3}	1.22×10^{2}	1.14×10^{-12}	36	1.11×10^{-12}
	1.0×10^{-8}	1.59×10^{3}	1.22×10^{2}	1.43×10^{-12}	36	1.04×10^{-12}
	1.0×10^{-9}	1.59×10^{3}	1.22×10^{2}	1.35×10^{-12}	36	1.37×10^{-12}

and a Cholesky factorization. Unfortunately, QR factorizations are more expensive and while strategies for computing incomplete orthogonal factorizations for use in building preconditioners have been proposed (see, for instance, [2–4, 27, 30, 34, 47]), the only available software is the MIQR package of Li and Saad [30] (probably because developing high quality implementations is non-trivial). In their study of preconditioners for LS problems, Gould and Scott [19, 20] found that MIQR generally

Table 6 Convergence results for problems sctap1-2r with $\omega = 1.0 \times 10^{-8}$ and stormg2-8_20 with $\omega = 1.0 \times 10^{-6}$. *tol* and iters are the convergence tolerance and the iteration count for GMRES

tol	Sctap1-2r		Stormg2-8_20	
	iters	$\|r_c\|$	iters	$\|r_c\|$
1.0×10^{-6}	2	1.669×10^{-6}	130	1.423×10^{-7}
1.0×10^{-7}	3	6.046×10^{-8}	134	5.364×10^{-9}
1.0×10^{-8}	3	6.046×10^{-8}	141	1.434×10^{-9}
1.0×10^{-9}	4	1.897×10^{-8}	146	1.246×10^{-10}
1.0×10^{-10}	4	1.897×10^{-8}	149	9.067×10^{-11}
1.0×10^{-11}	6	7.642×10^{-13}	156	1.314×10^{-10}
1.0×10^{-12}	6	7.642×10^{-13}	161	5.220×10^{-11}
1.0×10^{-13}	6	7.642×10^{-13}	190	4.972×10^{-12}
1.0×10^{-14}	7	9.136×10^{-13}	217	4.974×10^{-12}

performed less well than incomplete Cholesky factorization preconditioners and so is not considered here.

We have made the implicit assumption that A is sparse. However, it is straightforward to extend the augmented system-based approaches to the more general case that A contains rows that are dense. For example, if A is permuted and partitioned as

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix},$$

where A_1 is sparse and A_2 is dense, then using a conformal partitioning of y_s and of b, (4.7) can be replaced by the augmented system

$$\begin{pmatrix} -H_1(\omega) & C_d^T \\ C_d & \omega^2 I \end{pmatrix} \begin{pmatrix} x \\ y_d \end{pmatrix} = \begin{pmatrix} -A_1^T b_1 \\ d \end{pmatrix}$$

with

$$H_1(\omega) = A_1^T A_1 + \omega^2 I, \quad y_d = \begin{pmatrix} y_c \\ y_2 \end{pmatrix}, \quad C_d = \begin{pmatrix} C \\ \omega A_2 \end{pmatrix}, \quad d = \begin{pmatrix} d \\ \omega b_2 \end{pmatrix}.$$

Finally, we remark that, if we use the 3-block form (4.6) then we can follow [43], which in turn generalises the work of Carson, Higham and Pranesh [11], and obtain an augmented system approach with multi-precision refinement. This has the potential to reduce the computational cost in terms of time and/or memory, thus allowing larger problems to be solved.

5 Conclusions

We have considered a number of approaches for solving large-scale LSE problems in which the constraints may be dense. Our main findings can be summarized as follows:

- The classical null-space method relies on computing a null-space basis matrix Z for the "wide" constraint matrix C such that $Z^T A^T A Z$ is sparse. In recent work [44], we proposed how this can be achieved using a method based on a QR factorization of C with threshold pivoting. This is not straightforward to implement. Furthermore, our numerical experiments show that, in some cases, the norm $\|r_c\|$ of the constraints residual can be larger than for other approaches considered in this study. Thus, although in some contexts null-space approaches are popular, we do not recommend the strategy of [44] for LSE problems.
- The direct elimination approach couples the constraint matrix and the LS matrix, leading to a sparse-dense transformed least squares problem. Existing direct or iterative methods can be used to solve the transformed problem and our experiments found the computed constraint residuals are small. The approach can be used for problems for which A (as well as C) contains a small number of dense rows. A weakness is that, if solving a sequence of problems in which either A or C is fixed, the coupling of the two blocks in the solution process means that it must be restarted. Furthermore, the number of dense rows in the transformed problem can be relatively large, making it expensive to solve.
- There are several options for using an augmented system formulation. This can be solved using standard building blocks, such as a sparse QR factorization, a sparse symmetric indefinite linear solver, or a block sparse Cholesky factorization. An attraction of each of these is that existing "black box" solvers can be exploited, thereby greatly reducing the effort required in developing robust and efficient implementations. The augmented system formulation can be generalised to handle dense rows in A and offers the potential for mixed-precision computation. Moreover, an incomplete Cholesky factorization can be used as a preconditioner with a Krylov subspace solver.
- In the case of a series of LSE problems in which only the constraints change, both the null-space and direct elimination approaches have the disadvantage that the computation must be redone for each new set of constraints. For the augmented system approaches, a significant amount of work can be reused from the first problem in the sequence when solving subsequent problems.

Finally, we observe that there is a lack of iterative methods and preconditioners that can be used to extend the size of LSE problems that can be solved. We have shown that using an incomplete factorization within a block factorization of an augmented system can be effective, but most current incomplete factorizations that result in efficient preconditioners are serial in nature and not able to tackle extremely large problems (but see [1, 25] for novel approaches that are designed to exploit parallelism). Addressing the lack of iterative approaches is a challenging subject for future work.

Acknowledgements We are grateful to Professor Michael Saunders and an anonymous reviewer for their constructive comments that have led to many improvements in the presentation of this paper.

References

1. Anzt, H., Chow, E., Dongarra, J.: ParILUT-a new parallel threshold ILU factorization. SIAM J. on Scientific Computing **40**(4), C503–C519 (2018)
2. Bai, Z.-Z., Duff, I.S., Wathen, A.J.: A class of incomplete orthogonal factorization methods. I: Methods and theories. BIT Numer. Math. **41**(1), 53–70 (2001)
3. Bai, Z.-Z., Duff, Iain S., Yin, J.-F.: Numerical study on incomplete orthogonal factorization preconditioners. J. Comput. Appl. Math. **226**(1), 22–41 (2009)
4. Bai, Z.-Z., Yin, J.-F.: Modified incomplete orthogonal factorization methods using Givens rotations. Computing **86**(1), 53–69 (2009)
5. Barlow, J.L., Handy, S.L.: The direct solution of weighted and equality constrained least-squares problems. SIAM J. on Scientific Computing **9**(4), 704–716 (1988)
6. Björck, Å.: A general updating algorithm for constrained linear least squares problems. SIAM J. on Scientific and Statistical Computing **5**(2), 394–402 (1984)
7. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
8. Björck, Å.: Numerical Methods in Matrix Computations, volume 59 of Texts in Applied Mathematics. Springer, Cham (2015)
9. Björck, Å., Duff, I.S.: A direct method for the solution of sparse linear least squares problems. Linear Algebra Appl. **34**, 43–67 (1980)
10. Björck, Å., Golub, G.: ALGOL Programming, Contribution No. 22: Iterative refinement of linear least square solutions by Householder transformation. BIT Numer. Math. **7**, 322–337 (1967)
11. Carson, E., Higham, N., Pranesh, S.: Three-precision GMRES-based iterative refinement for least squares problems. SIAM J. on Scientific Computing **42**(6), A4063–A4083 (2020)
12. Coleman, T.F., Pothen, A.: The null space problem. I. Complexity. SIAM J. on Algebraic and Discrete Methods **7**(4), 527–537 (1986)
13. Damm, T., Stahl, D.: Linear least squares problems with additional constraints and an application to scattered data approximation. Linear Algebra Appl. **439**(4), 933–943 (2013)
14. Davis, T.A.: Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. ACM Transactions on Mathematical Software **38**(1), 8:1-8:22 (2011)
15. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. ACM Transactions on Mathematical Software **38**(1), 1–28 (2011)
16. Farebrother, R.W.: Visualizing Statistical Models and Concepts, volume 166 of Statistics: Textbooks and Monographs. Marcel Dekker, Inc., New York (2002)
17. Golub, G.H.: Numerical methods for solving least squares problems. Numer. Math. **7**, 206–216 (1965)
18. Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. The Johns Hopkins University Press, Baltimore and London (1996)
19. Gould, N.I.M., Scott, J.A.: The state-of-the-art of preconditioners for sparse linear least squares problems: the complete results. Technical Report RAL-TR-2015-009, Rutherford Appleton Laboratory (2015)
20. Gould, N.I.M., Scott, J.A.: The state-of-the-art of preconditioners for sparse linear least squares problems. ACM Transactions on Mathematical Software **43**(4), 36:1–35 (2017)
21. Hanson, R.J., Lawson, C.L.: Extensions and applications of the Householder algorithm for solving linear least squares problems. Math. Comput. **23**, 787–812 (1969)
22. Heath, M.T.: Some extensions of an algorithm for sparse linear least squares problems. SIAM J. on Scientific and Statistical Computing **3**(2), 223–237 (1982)

23. Hogg, J.D., Reid, J.K., Scott, J.A.: Design of a multicore sparse Cholesky factorization using DAGs. SIAM J. on Scientific Computing **32**, 3627–3649 (2010)
24. Hogg, J.D., Scott, J.A.: New parallel sparse direct solvers for multicore archiectures. Algorithms **6**, 702–725 (2013)
25. Hook, J., Scott, J., Tisseur, F., Hogg, J.: A max-plus approach to incomplete Cholesky factorization preconditioners. SIAM J. on Scientific Computing **40**(4), A1987–A2004 (2018)
26. HSL. A collection of Fortran codes for large-scale scientific computation (2018). http://www.hsl.rl.ac.uk
27. Jennings, A., Ajiz, M.A.: Incomplete methods for solving $A^T Ax = b$. SIAM J. on Scientific and Statistical Computing **5**(4), 978–987 (1984)
28. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. Prentice-Hall, Inc., Englewood Cliffs, N.J. (1974). Prentice-Hall Series in Automatic Computation
29. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems, volume 15 of Classics in Applied Mathematics. SIAM, Philadelphia (1995). Revised reprint of the 1974 original
30. Li, N., Saad, Y.: MIQR: A multilevel incomplete QR preconditioner for large sparse least-squares problems. SIAM J. on Matrix Analysis and Applications **28**(2), 524–550 (2006)
31. Murtagh, B.A., Saunders, M.A.: MINOS 5.51 "User's Guide". Technical Report SOL-83-20, Systems Optimization Laboratory, Dept. of Operations Research, Stanford Univ. (2003)
32. Nash, S.G., Sofer, A.: Preconditioning reduced matrices. SIAM J. on Matrix Analysis and Applications **17**(1), 47–68 (1996)
33. Paige, C.C., Saunders, M.A.: Solution of sparse indefinite systems of linear equations. SIAM J. on Numerical Analysis **12**(4), 617–629 (1975)
34. Papadopoulus, A.T., Duff, I.S., Wathen, A.J.: A class of incomplete orthogonal factorization methods. II: Implementation and results. BIT Numer. Math. **45**(1), 159–179 (2005)
35. Pisinger, G., Zimmermann, A.: Bivariate least squares approximation with linear constraints. BIT Numer. Math. **47**(2), 427–439 (2007)
36. Powell, M.J.D., Reid, J.K.: On applying Householder transformations to linear least squares problems. In: Information Processing 68 (Proc. IFIP Congress, Edinburgh, 1968), Vol. 1: Mathematics, Software, pages 122–126. North-Holland, Amsterdam (1969)
37. Saad, Y., Schultz, M.H.: GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. on Scientific and Statistical Computing **7**, 856–869 (1986)
38. Saunders, M.A.: Solution of sparse rectangular systems using LSQR and CRAIG. BIT Numer. Math. **35**(4), 588–604 (1995)
39. Schittkowski, K., Stoer, J.: A factorization method for the solution of constrained linear least squares problems allowing subsequent data changes. Numerische Mathematik **31**(4), 431–463 (1978/79)
40. Scott, J.A., Tůma, M.: Solving mixed sparse-dense linear least-squares problems by preconditioned iterative methods. SIAM J on Scientific Computing **39**(6), A2422–A2437 (2017)
41. Scott, J.A., Tůma, M.: A Schur complement approach to preconditioning sparse least-squares problems with some dense rows. Numerical Algorithms **79**(4), 1147–1168 (2018). https://doi.org/10.1007/s11075-018-0478-2
42. Scott, J.A., Tůma, M.: Strengths and limitations of stretching for least-squares problems with some dense rows. ACM Transactions on Mathematical Software **47**(1), 1:1-25 (2021)
43. Scott, J.A., Tůma, M.: A computational study of using black-box QR solvers for large-scale sparse-dense linear least squares problems. ACM Transactions on Mathematical Software **48**(1), 5:1-24 (2022)
44. Scott, J.A., Tůma, M.: A null-space approach for large-scale symmetric saddle point systems with a small and non zero (2,2) block. Numerical Algorithms, 2022. published online
45. Stoer, J.: On the numerical solution of constrained least-squares problems. SIAM J. on Numerical Analysis **8**, 382–411 (1971)
46. Van Loan, C.: On the method of weighting for equality-constrained least-squares problems. SIAM J. on Numerical Analysis **22**(5), 851–864 (1985)
47. Wang, X., Gallivan, K.A., Bramley, R.: CIMGS: an incomplete orthogonal factorization preconditioner. SIAM J. on Scientific Computing **18**(2), 516–536 (1997)
48. Zhdanov, A.I.: The method of augmented regularized normal equations. Comput. Math. Math. Phys. **52**(2), 194–197 (2012)

49. Zhdanov, A.I., Gogoleva, S.Y.: Solving least squares problems with equality constraints based on augmented regularized normal equations. Applied Mathematics E-Notes **15**, 218–224 (2015)
50. Zhu, Y., Li, X.R.: Recursive least squares with linear constraints. Commun. Inf. Syst. **7**(3), 287–311 (2007)