Preconditioners for rank deficient least squares problems ⋆

J. Cerdán[1], D. Guerrero[2], J. Marín[1], J. Mas[1]

Abstract

In this paper we present a method for computing sparse preconditioners for iteratively solving rank deficient least squares problems (LS) by the LSMR method. The main idea of the method proposed is to update an incomplete factorization computed for a regularized problem to recover the solution of the original one. The numerical experiments for a wide set of matrices arising from different science and engineering applications show that the preconditioner proposed, in most cases, can be successfully applied to accelerate the convergence of the iterative Krylov subspace method.

Keywords: Iterative methods, rank deficient, sparse linear systems, preconditioning, linear least squares problems.

2000 MSC: , 65F08, 65F10, 65F50, 65N22

1. Introduction

Linear least squares (LS) problems arise in many large-scale applications of the science and engineering as neural networks, linear programming, exploration seismology or image processing, among others. The LS problem considered is

⋆This work was supported by the Spanish Ministerio de Economía, Industria y Competitividad under grants MTM2017-85669-P and MTM2017-90682-REDT.

Email addresses: jcerdan@imm.upv.es (J. Cerdán), danguefl@doctor.upv.es (D. Guerrero), jmarinma@imm.upv.es (J. Marín), jmasm@imm.upv.es (J. Mas)

[1]Institut de Matemàtica Multidisciplinar, Universitat Politècnica de València, 46022 València, España.

[2]Departamento de Ciencias Matemáticas, Universidad Pedagógica Nacional Francisco Morazán, Honduras.

formulated as

$$\min_x \|b - Ax\|_2, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ is large and sparse and $b \in \mathbb{R}^m$. This problem can be also formulated with the mathematically equivalent $n \times n$ normal equations system:

$$A^T Ax = A^T b. \tag{2}$$

Two types of methods are usually used to solve these linear systems, direct and iterative methods. Direct methods, in spite of their robustness, require the computation of an explicit factorization of the coefficient matrix of the linear system, that implies large computational time and memory storage. In contrast, iterative Krylov subspace methods may be preferred when the system matrix is large and sparse because they often are less demanding in memory requirements than their direct counterparts. In this case, Equation (2) is solved iteratively using conjugate gradient like methods. Basically, these methods implicitly apply the conjugate gradient or minimal residual method to the normal equations.

The CGLS [1] and LSQR methods implicitly apply the conjugate gradient to the normal equations. In exact arithmetic, both methods generate the same sequence of approximation. The LSQR was presented in [2, 3]. It is based on the Golub-Kahan bidiagonalization developed in [4]. LSQR has the property of reducing $||r_k||_2$ monotonically, where $r_k = b - Ax_k$ is the residual for the approximate solution x_k.

Another alternative is the least squares minimal residual method (LSMR) presented in [5]. LSMR, as LSQR, also uses the Golub-Kahan bidiagonalization of A. But in contrast to the LSQR, LSMR is equivalent to MINRES applied to the normal equations. One of the characteristics of the LSMR method is that the norm of the residual $||r_k||_2$ decreases monotonically, and is never very far behind the corresponding value for LSQR. Hence, although LSQR and LSMR ultimately converge to similar points, it is safer to use LSMR in situations where the solver must be terminated early.

The successful application of an iterative method, often needs a good pre-

conditioner to achieve fast convergence rates. When the matrix A has full rank, typically an incomplete Cholesky factorization (IC) of the symmetric and positive definite matrix $C = A^T A$ is used as preconditioner. An extended review of a variety of preconditioners already available in the repositories can be found in [6]. For instance, diagonal preconditioning, limited memory incomplete Cholesky factorization developed for the HSL mathemathical software library, [7], the Multilevel Incomplete QR (MIQR) factorization [8], the Robust Incomplete Factorization [9], the BA-GMRES for solving least squares problems [10] and incomplete Cholesky based on BIF preconditioner [11]. If the matrix A is rank deficient then, C is a semidefinite positive matrix and the Cholesky factorization may suffer breakdown because negative or zero pivots are encountered. Thus, rank deficient LS problems are in general much more harder to solve.

Basically, there are two types of approaches for solving this case iteratively. The first one consists of computing an incomplete factorization of a regularized matrix which can be used as a preconditioner for the original LS problem. The second type computes the solution of a mathematically equivalent augmented linear system of order $m+n$ [1, 12]. Both strategies are extensively analyzed in [12]. In this work, IC preconditioners for the regularized problem are compared with a direct solver used as preconditioner. It is concluded that the IC factorization is less efficient. However, an advantage is that the IC factors are considerably sparse, giving it the potential to be used successfully for much larger problems. The technique proposed in this work follows this line of research. It consists of updating an IC factorization computed for the regularized system which is used as preconditioner for solving the original problem iteratively.

The paper is organized as follows. In Section 2, we describe the bordering technique used to update an existing preconditioner by using an equivalent augmented system. In Section 3, we describe the test environment, and present the set of problems used for the numerical experiments. Then, we report the results of the tests in Section 4. Finally, the main conclusions are presented.

2. Updated preconditioner method

When the matrix A in (1) is rank deficient, one of the approaches for solving the LS problem is based on the computation of a Cholesky factorization of the normal equations associated to the regularized matrix

$$\begin{bmatrix} A \\ \alpha^{1/2} I \end{bmatrix}, \tag{3}$$

which are given by

$$C_\alpha = A^T A + \alpha I. \tag{4}$$

The shift α is known as Tikhonov regularization parameter. If α is choosen large enough the computation of an IC for the matrix C_α can be done easily and without breakdowns. On the other hand, since the final purpose is to use this incomplete factorization as a preconditioner for the original (unregularized) linear system, the parameter α should be chosen as small as possible. Both requirements make difficult the choice of the appropriate α. In practice, one starts with a very small value for α, typically $\mathcal{O}(10^{-14})$, and if a breakdown occurs then α is increased successively until a successful incomplete factorization is computed. This iterative process may be expensive since multiple incomplete factorizations must be computed before solving the least squares problem. Moreover, the density of the preconditioners often is quite large. One of the goals of this work is to avoid this process by choosing a large enough value of the shift α, and then update the preconditioner in order to obtain a better approximation of the original problem. Moreover, with larger values of α the regularized matrix is more diagonal dominant and the computed preconditioners tend to be more sparse. The technique used is based on the work presented in [13] in which the authors study how to update a preconditioner for LS problems when the linear system is modified by adding or removing equations.

The idea is to compute a preconditioner for the regularized matrix C_α in (4), with α large enough, and then update the preconditioner for the original problem. Consider the matrix

$$C_\alpha - \beta I \tag{5}$$

which is an update of the shifted matrix C_α in (4). Clearly, the closer β is to α, the closer this update is to the normal equations $A^T A$.

Our technique consists of updating an incomplete Cholesky factorization obtained for C_α. It relies on the relations that one can stablish between the augmented matrix

$$\begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}, \tag{6}$$

and the matrix in (5). Observe that

$$C_\alpha - \beta I = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix} \begin{bmatrix} I \\ -\beta^{1/2} I \end{bmatrix}, \tag{7}$$

and

$$(C_\alpha - \beta I)^{-1} = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix}. \tag{8}$$

Thus, an IC factorization computed for the augmented matrix (6) can be used to approximate the matrix $C_\alpha - \beta I$, and its inverse. Hence, it can be used as a preconditioner for the original normal equations if it is applied properly. Note that the modification introduced by the update in equation (5), makes the choice of the shift α less restrictive since with the update we somehow neutralize the bad effect of the regularization, that is, the computation of a preconditioner that may be far from the original problem. The only condition needed now is to select a large enough value to avoid breakdown during the computation of an IC factorization. We found experimentally that a value of $\alpha = 1$ was good for the majority of the problems tested.

With respect to the choice of β we recall that the ideal choice should be $\alpha = \beta$ in order to obtain the closest preconditioner to the original problem. But, since we are dealing with rank defficient LS problems, i.e., rank $A = k < n$, theoretically one should choose $\beta \neq \alpha$. Otherwise, the square augmented matrix in (6) is singular. To see this, let us do a symmetric permutation to the

augmented matrix as follows

$$\begin{bmatrix} O & I \\ I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix} \begin{bmatrix} O & I \\ I & O \end{bmatrix} = \begin{bmatrix} I & \beta^{1/2} I \\ \beta^{1/2} I & C_\alpha \end{bmatrix}.$$

After eliminating the left bottom block by Gaussian elimination we obtain

$$\begin{bmatrix} I & \beta^{1/2} I \\ 0 & C_\alpha - \beta I \end{bmatrix} = \begin{bmatrix} I & \beta^{1/2} I \\ 0 & A^T A + (\alpha - \beta) I \end{bmatrix}, \tag{9}$$

which has no full rank if $\beta = \alpha$. The interval $0 \leq \beta < \alpha$ may be a good choice from this point of view. However, we found experimentally that taking β equals to α did not produce breakdown during the preconditioner computation and often it performed the best. To illustrate this, Figure 1 shows the number of iterations and time for different values of β.

A possible explanation is that to obtain the factorization (10) which is used as preconditioner, we are computing incomplete factorization of C_α and also for an approximated Schur complement $R \approx I - \beta C_\alpha^{-1}$, (see Algorithm 1). As a consequence, block (2,2) in equation (9) for $\alpha = \beta$ is not exactly $A^T A$ but $A^T A + E$ that has full rank.

2.1. Preconditioner computation

The preconditioner is obtained from the incomplete block Cholesky factorization of the augmented matrix in (6) given by

$$\begin{pmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{pmatrix} = \begin{pmatrix} L_\alpha & 0 \\ \beta^{1/2} L_\alpha^{-T} & L_R \end{pmatrix} \begin{pmatrix} L_\alpha^T & \beta^{1/2} L_\alpha^{-1} \\ 0 & L_R^T \end{pmatrix} \tag{10}$$

where L_α is the incomplete Cholesky factor of C_α and L_R is the incomplete Cholesky factor of the Schur complement of C_α in the augmented matrix, $R = I - \beta L_\alpha^{-T} L_\alpha^{-1}$. The preconditioner is computed in four steps, which are summarized in Algorithm 1.

To keep the preconditioner sparse, the amount of fill-in may be limited by dropping small elements in steps 2 and 3, so an approximate Schur complement it is computed. Besides, in step 4 either an exact or an incomplete factorization of the Schur complement R may be computed depending on its size.

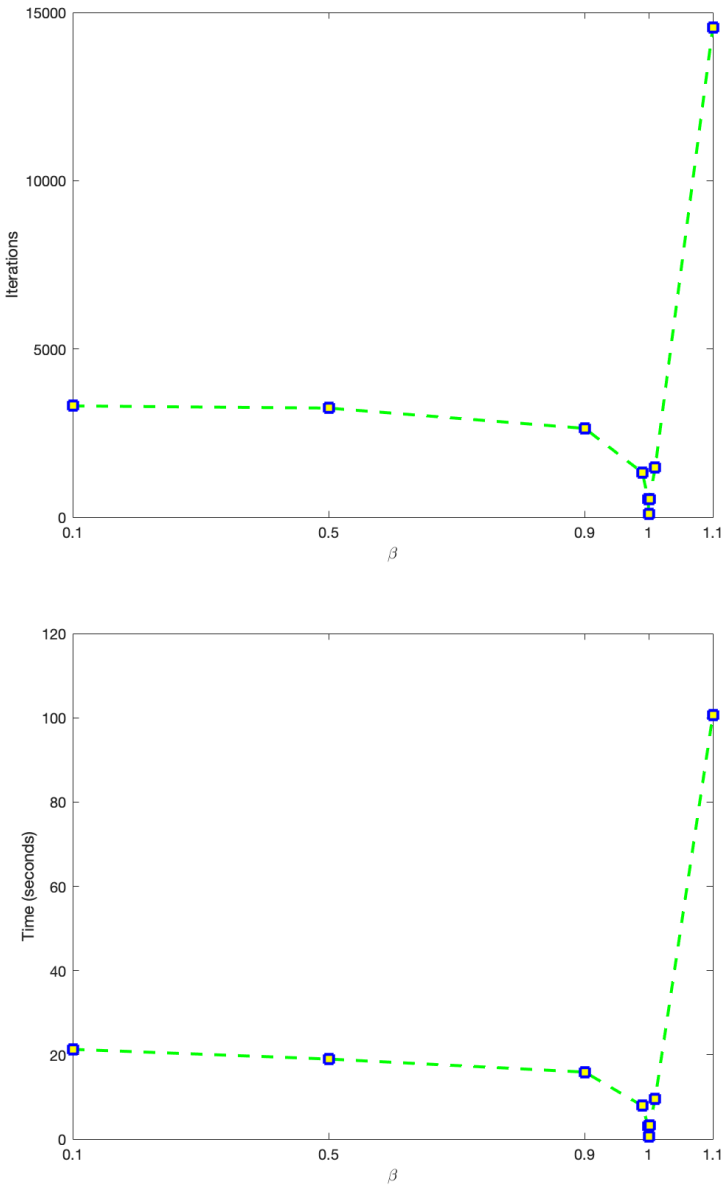

Figure 1: DBIR1 matrix. Effect of different values of β in the number of iterations and time for $\alpha = 1$.

Algorithm 1 Preconditioner update computation

1. Compute $L_\alpha L_\alpha^T \approx C_\alpha$.
2. Compute $T = \beta^{1/2} L_\alpha^{-1}$.
3. Compute $R = I - T^T T$.
4. Compute $L_R L_R^T \approx R$.

2.2. Preconditioner application

The preconditioning step for a Krylov subspace iterative method involves the solution of systems of the form $Ms = r$ where M is the preconditioner and r is the residual. Thus, the preconditioning strategy proposed computes the preconditioned residual by applying equation (8) with an incomplete factorization of the augmented matrix. That is, the preconditioned residual s is given by

$$s = \begin{bmatrix} I & O \end{bmatrix} \begin{bmatrix} C_\alpha & \beta^{1/2} I \\ \beta^{1/2} I & I \end{bmatrix}^{-1} \begin{bmatrix} I \\ O \end{bmatrix} r,$$

and it is computed from the solution of the block linear system

$$\begin{pmatrix} L_\alpha & 0 \\ \beta^{1/2} L_\alpha^{-T} & L_R \end{pmatrix} \begin{pmatrix} L_\alpha^T & \beta^{1/2} L_\alpha^{-1} \\ 0 & L_R^T \end{pmatrix} \begin{pmatrix} s \\ s_1 \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}.$$

The preconditioning step is summarized in Algorithm 2.

Algorithm 2 Preconditioner update application

Input: β, matrices L_α, L_R and residual vector r.

Output: Preconditioned vector s

1. $s \leftarrow L_\alpha^{-T}(L_\alpha^{-1} r)$.
2. $s_R \leftarrow L_R^{-T}(L_R^{-1} \bar{r})$.
3. $s \leftarrow s + \beta L_\alpha^{-T}(L_\alpha^{-1} s_R)$.

It will be referenced as updated preconditioner method (UPD). In this algorithm, steps 2 and 3 represent the extra cost in the application of the preconditioner with respect to standard preconditioning with the IC factorization of the regularized matrix, i.e., the non-updated preconditioner. We recall that

the inverses of the triangular factors are applied by solving the corresponding triangular systems. If L_α and L_R are kept sparse, the additional cost is small and can be amortized even for moderate reductions on the number of iterations, as pointed out in [14].

3. Numerical environment

The experiments were done with MATLAB version 2016a running on an Intel 5 CPU with 8 Gb of RAM. For the solution of each problem we set a limit of 600 seconds and 50,000 iterations for the total CPU time and number of iterations, respectively. As recommended in MATLAB's documentation, CPU times reported are the mean value of 10 successive runs of the experiment performed after 3 initial runs that were discarded. The maximum standard deviation observed relative to the mean value for the different runs was insignificant.

Table 1 shows the set of tested matrices from the Florida Sparse Matrix Collection [15], arising in different areas of scientific computing. The matrices were cleaned by removing the null rows and columns before solving the LS problem. The number of rows and columns, number of nonzeros (nnz) and nullity of the matrix (estimated null space rank) are also reported.

If a matrix has less rows than columns, then it is transposed. Each matrix was permuted using the MATLAB's function **dmperm**() that obtains the Dulmage-Mendelsohn decomposition [16]. This decomposition estimates an upper bound of the structural rank of the matrices, that allows for the approximation of their nullity. The values obtained are in agreement with the ones reported in [6]. The columns of the matrix corresponding to the normal equations $C = A^T A$ were normalised by their 2-norm. As result, the regularized normal equations matrix is

$$C_\alpha = SPCP^T S^T + \alpha I.$$

An IC factorization $L_\alpha L_\alpha^T$ of C_α was computed with the MATLAB's function **ichol**(). With respect to the Schur complement R in Algorithm 1, small elements

Matrix	m	n	nnz	nullity	Application
BAXTER	27441	30733	111576	3055	Linear programming
DBIR1	18804	45775	1077025	2	Linear programming
DBIR2	18906	45877	1158159	2	Linear programming
NSCT1	22901	37461	678739	1	Linear programming
NSCT2	23003	37563	697738	1	Linear programming
beaflw	492	500	53403	32	Economic
Pd_rhs	5804	4371	6323	3	Counter-example
162bit	3606	3476	37118	16	Combinatorial
176bit	7441	7150	82270	40	Combinatorial
192bit	13691	13093	154303	87	Combinatorial
208bit	24430	23191	299756	210	Combinatorial
wheel_601	902103	723605	2170814	600	Combinatorial
12month1	12471	872622	22624727	53	Bipartite graph
ND_actors	383640	127823	1470404	13061	Bipartite graph
IMDB	303617	896302	3782463	53101	Bipartite graph
Maragal_6	21251	10144	537694	92	Least squares
Maragal_7	46845	26525	1200537	659	Least squares
Maragal_8	33093	60845	1308415	14637	Least squares
mri1	65536	114637	589824	1019	graphics/vision
mri2	63240	104597	569160	14919	graphics/vision
tomographic1	142752	1014301	11537419	3700	graphics/vision

Table 1: Set of tested matrices

were dropped in step 2 before computing step 3, and finally an incomplete LU factorization using MATLAB's function **ilu**() was computed in step 4. For the updated preconditioner, a value of $\alpha = \beta = 1$ was used for all the matrices, except for the matrices BAXTER and BEAFLW for which a value of $\alpha = \beta = 10^{-3}$ was needed. We have found that in practice, choosing $\alpha = \beta$, has given the best results. The right-hand side b was the vector of all ones.

The LSMR method was used to solve the normal equations because its nice property mentioned in the Section 1, that is, the residual norm decreases monotonically. The Matlab's impletation of LSMR was downloaded from [17]. Since our method does not compute an explicit factorization of the normal equations, applying two side preconditioning is not possible as suggested by the authors. Following the ideas in [18] where a left preconditioned LSQR algorithm is derived, we implemented a left preconditioned version of the LSMR which is presented in Algorithm 3. The application of the preconditioner is done in the initialization and bidiagonalization steps.

Algorithm 3 Left preconditioned LSMR

1. Initialize

$$\beta_1 u_1 = b \qquad \tilde{p} = A^T u_1 \qquad v_1 = M^{-1}\tilde{p} \qquad \alpha_1 = (v_1, \tilde{p})^{1/2}$$

$$v_1 = v_1/\alpha_1 \qquad \bar{\alpha}_1 = \alpha_1 \qquad \bar{\zeta}_1 = \alpha_1\beta_1 \qquad \rho_0 = \bar{\rho}_0 = \bar{c}_0 = 1$$

$$\bar{s}_0 = 0 \qquad h_1 = v_1 \qquad \bar{h}_0 = x_0 = \overrightarrow{0}$$

2. For $k = 1, 2, 3, \ldots,$ until convergence Repeat steps 3-6:

3. Continue the bidiagonalization

$$\beta_{k+1} u_{k+1} = A v_k - \alpha_k u_k$$

$$\tilde{p} = A^T u_{k+1} - \beta_{k+1}\tilde{p}, \quad v_{i+1} = M^{-1}\tilde{p}, \quad \alpha_{i+1} = (v_{i+1}, \tilde{p})^{1/2}, \quad \tilde{p} = \tilde{p}/\alpha_{i+1},$$

$$v_{i+1} = v_{i+1}/\alpha_{i+1}$$

4. (Construct and apply rotation $Q_{k,k+1}$)

$$\rho_k = (\bar{\alpha}_k^2 + \beta_{k+1}^2)^{1/2}$$

$$c_k = \bar{\alpha}_k/\rho_k \qquad s_k = \beta_{k+1}/\rho_k$$

$$\theta_{k+1} = s_k\alpha_{k+1} \qquad \bar{\alpha}_{k+1} = c_k\alpha_{k+1}$$

5. (Construct and apply rotation $\bar{Q}_{k,k+1}$)

$$\bar{\theta}_k = \bar{s}_{k-1}\rho_k \qquad \bar{\rho}_k = ((\bar{c}_{k-1}\rho_k)^2 + \theta_{k+1}^2)^{1/2}$$

$$\bar{c}_k = \bar{c}_{k-1}\rho_k/\bar{\rho}_k \qquad \bar{s}_k = \theta_{k+1}/\bar{\rho}_k$$

$$\zeta_k = \bar{c}_k\bar{\zeta}_k \qquad \bar{\zeta}_{k+1} = -\bar{s}_k\bar{\zeta}_k$$

6. (Update h, $\bar{h}$ and x)

$$\bar{h}_k = h_k - (\bar{\theta}_k\rho_k/(\rho_{k-1}\bar{\rho}_{k-1}))\bar{h}_{k-1}$$

$$x_k = x_{k-1} + (\zeta/(\rho_k\bar{\rho}_k))\bar{h}_k$$

$$h_{k+1} = v_{k+1} - (\theta_{k+1}/\rho_k)h_k$$

4. Numerical experiments

In this section we study the numerical performance of the preconditioner update method proposed. The method has been compared with an IC factorization of the regularized matrix C_α. Before analysing the performance of the

preconditioner, we study the convergence criteria for the LSMR method and the choice of the Tikhonov regularization parameter.

4.1. On the convergence criteria and the choice of α

We have done an extensive study concerning the stopping criteria for the convergence of the LSMR method. Different convergence criteria are proposed in the bibliography.

FS: Fong and Saunders in [5] propose the following stopping rule

$$\frac{||A^T r_k||_2}{||A||_2||r_k||_2} < \epsilon. \tag{11}$$

GS: Gould and Scott in [12] proposed a different criterion defined

$$\frac{||A^T r_k||_2||r_0||_2}{||A^T r_0||_2||r_k||_2} < \epsilon, \tag{12}$$

that reduces to

$$\frac{||A^T r_k||_2||b||_2}{||A^T b||_2||r_k||_2} < \epsilon,$$

when the initial solution guess is $x_0 = 0$.

It can be easily observed the following relation between both criteria

$$\frac{||A^T r_k||_2}{||A||_2||r_k||_2} = \frac{||A^T r_k||_2||b||_2}{||A^T||_2||b||_2||r_k||_2} \leq \frac{||A^T r_k||_2||b||_2}{||A^T b||_2||r_k||_2}.$$

Thus, the LSMR method might reach convergence early with the FS criterion, as we will see below.

We remark that the convergence rule programmed by default in the LSMR implementation of Fong and Saunders depends on the preconditioner M applied, since it evaluates the norm $||(AM^{-1})^T r||_2$. To remove this dependency in the sense of Gould and Scott, we modify the FS criterion by computing instead the norm $||(A)^T r||_2$. Moreover, as the authors did in [12], we exclude the additional computational time needed to computed the corresponding residuals from the total solution time.

To study the effect of the stopping criteria on the convergence of the LSMR method, we consider for instance, the matrix $DBIR1$. The convergence tolerance was set to $\epsilon = 10^{-6}$. For the Tikhonov parameter α, values in the

interval $[0.01, 2]$ were considered. We note that, for very small values of α, the MATLAB's function **ichol**() produced very dense preconditioners. Therefore, to study the effect of α the IC factorization $L_\alpha L_\alpha^T$ of C_α was obtained with the MATLAB's function **ilu**() and drop tolerance 0.01. With this function sparser factorizations were obtained with a considerable reduction of the computational time.

Figure 2 shows the CPU time and the number of iterations that the LSMR method takes to converge with both stopping rules and both preconditioning strategies, i.e., an IC Cholesky factorization of C_α and the updated preconditioner. One can observe that the best results are obtained for values of α in the interval $[0.1, 1]$. But, an advantage of taking the largest value $\alpha = 1$ is that sparser incomplete Cholesky factors are often obtained since the regularized matrix is more diagonal dominant.

In general, we may recommend a value of $\alpha = 1$ since it peformed quite well for the majority of the problems tested. The results presented below are obtained with this value, with some exceptions.

With respect to the convergence criteria it is observed that the LSMR method with the FS stopping rule needed less number of iterations and computational time to converge for all the values of α.

Figure 3 shows the evolution of the FS and GS criteria for a fixed value of $\alpha = 1$ during the iterative solution process. As mentioned before, the FS rule converges in less iterations, specially with the non-updated preconditioner.

Figure 4 shows the evolution of the residuals $||r_k||_2$ and $||A^T r_k||_2$. It can be observed that the final residual norms are of the same order for both criteria, and a very small reduction on the norms is obtained with the GS rule at the cost of more iterations, that for some problems the difference can be considerably high.

Finally, Figure 5 compares the evolution of $||r_k||_2$ for the non-updated and updated preconditioners and both convergence criteria. It can be observed that the UPD preconditioner converges in less iterations than the non-updated one.

After this study, in the next subsection the results will be presented with

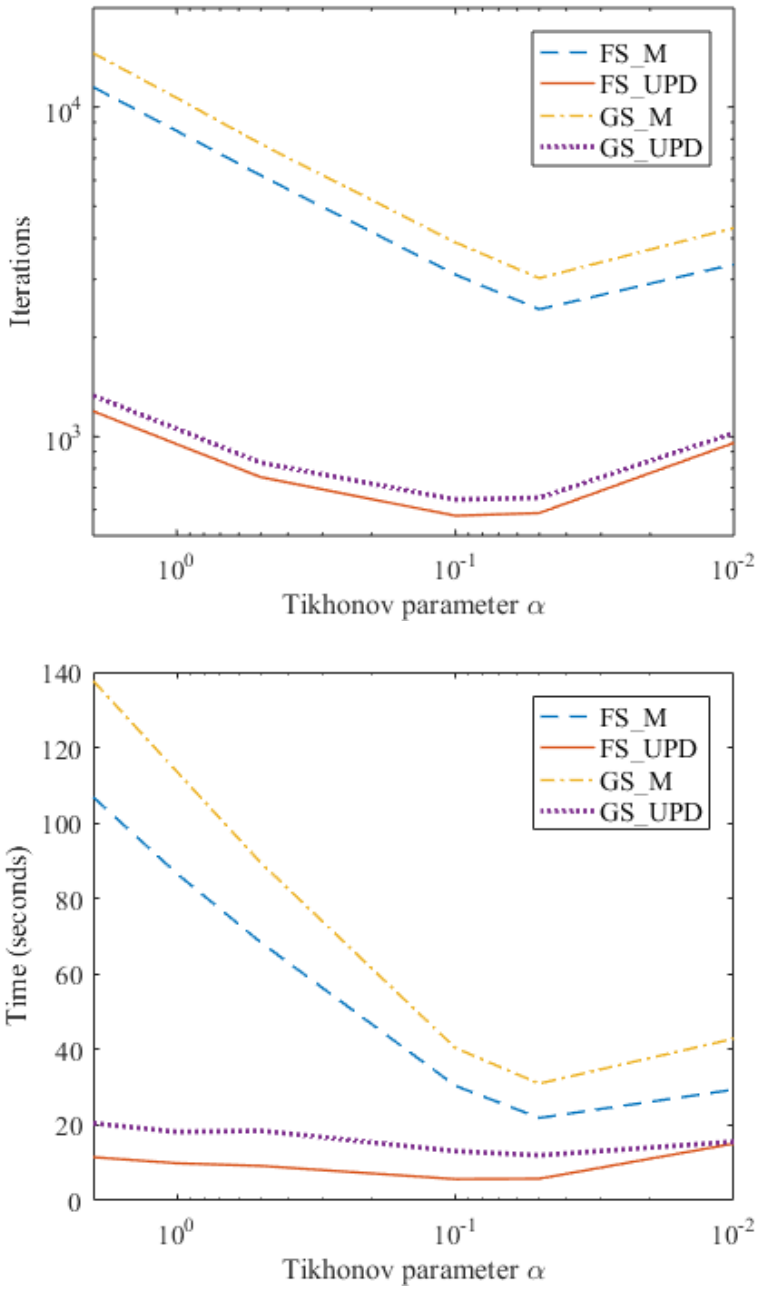

Figure 2: DBIR1 matrix. Number of iterations and total solution time, for the LSMR method with FS and GS criteria, and for the non updated (M) and updated (UPD) preconditioners, α in $[0.01, 2]$.

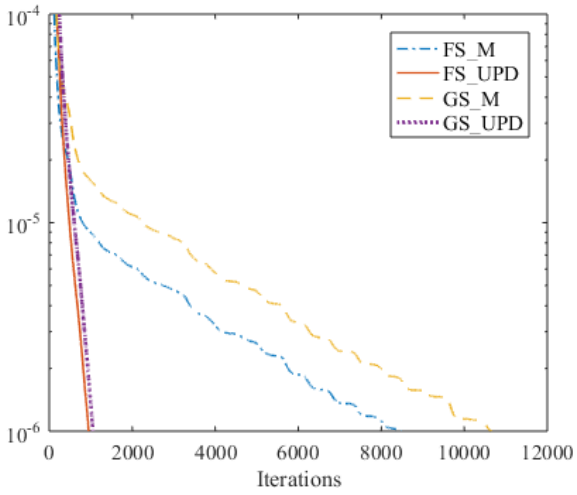

Figure 3: DBIR1 matrix. Evolution of the FS and GS criteria with respect to the number of iterations; y axis represents the value of the FS and GS criteria at each iteration.

the FS stopping criterion and with a value of $\alpha = 1$.

4.2. Results

Table 2 shows the results of the numerical experiments for the different matrices tested. In this table, $time_t$, $||r||_2$ and itn represent the total time (in seconds, including computation of the preconditioner), residual norm ($||b - Ax||_2$) and the number of iterations needed to converge, respectively. The number of nonzeros of the Cholesky factor L_α is represented by $nnz(L_\alpha)$, and $nnz(L_\alpha \wedge L_R) = nnz(L_\alpha) + nnz(L_R)$ represents the number the nonzero elements of the updated preconditioner. The column M corresponds to the results obtained with the IC factorization of C_α, while UPD corresponds to the ones obtained with the proposed updated preconditioning technique. The iterative method was stopped when the stopping rule FS was reduced to 10^{-6}. The IC factor L_α was calculated with drop tolerance 0.1, except for the matrices MRI1, MRI2 and BAXTER, BEAFLW for which a value of 0.2 and 10^{-5} were used, respectively.

The problems are classified into three blocks mainly taking into account the field of application. The best results for every problem, in total solution time

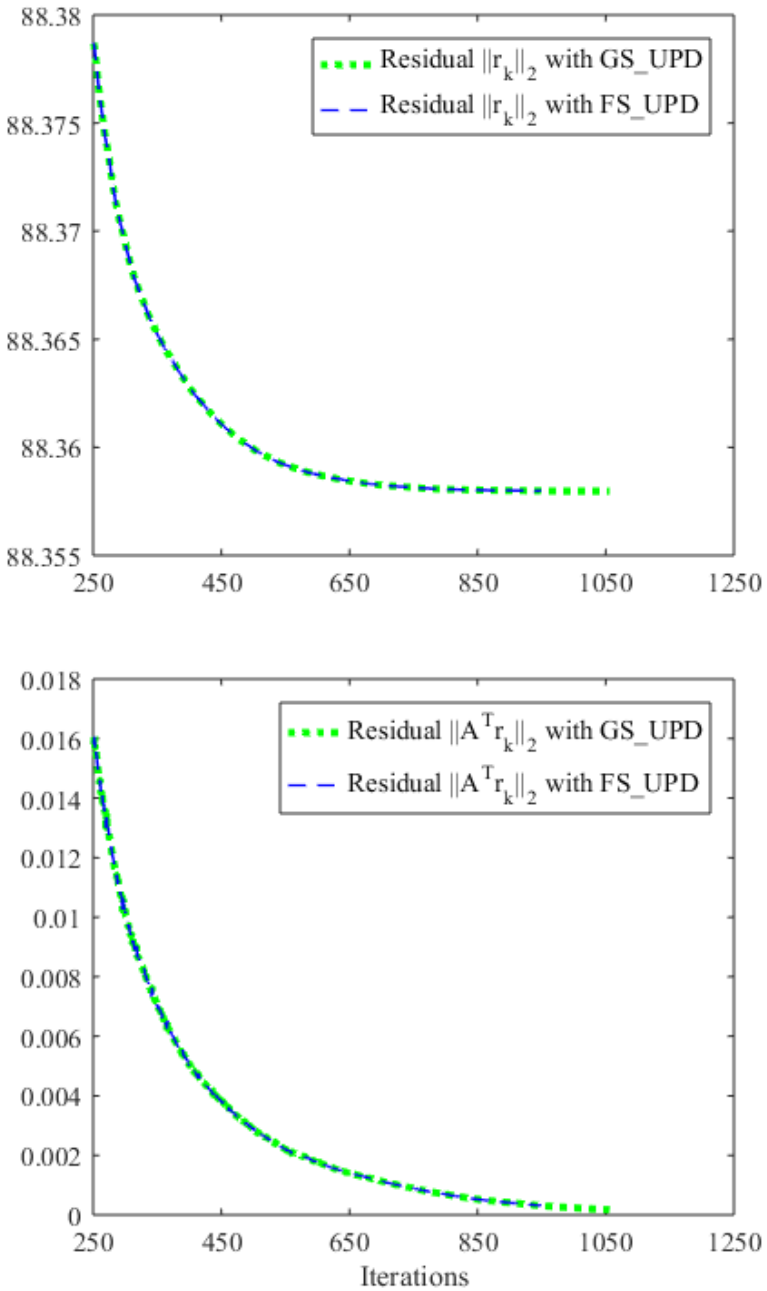

Figure 4: DBIR1 matrix. Evolution of $||r_k||_2$ and $||A^T r_k||_2$.

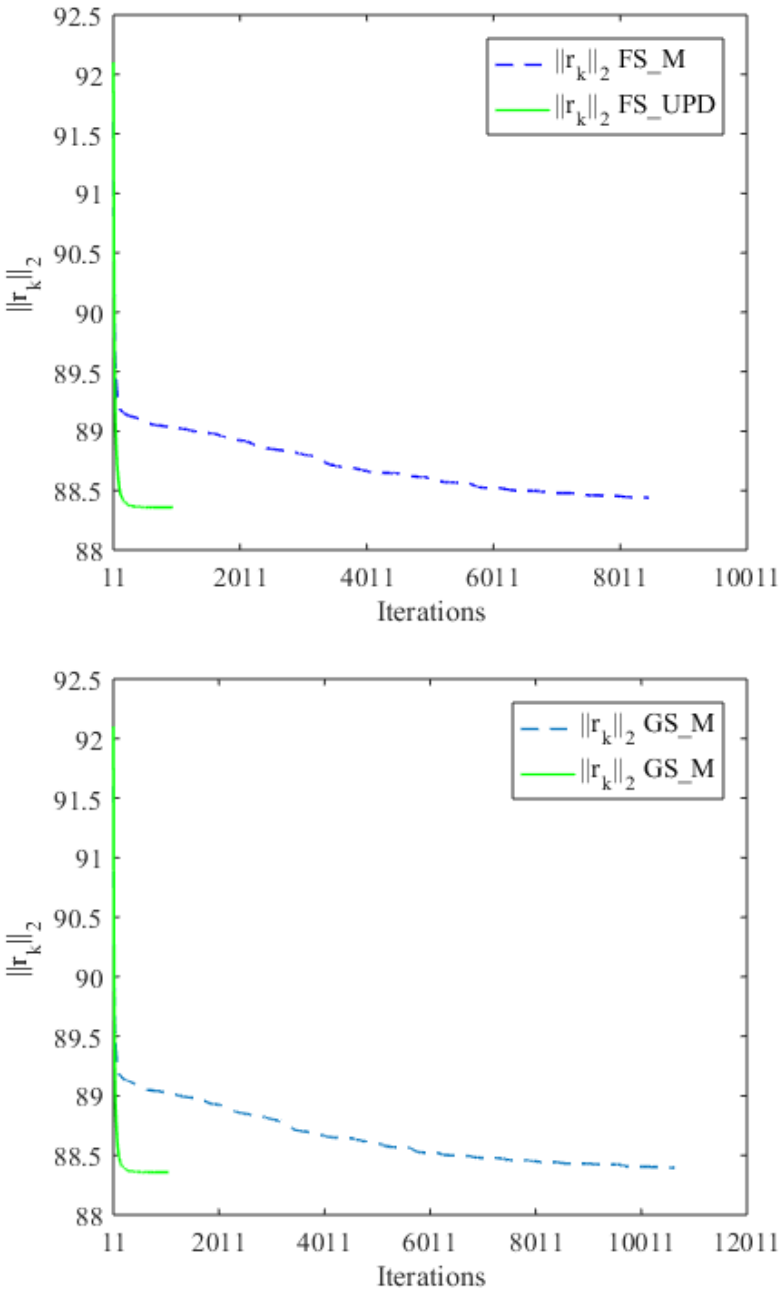

Figure 5: DBIR1 matrix. Evolution of $||r_k||_2$ for the non updated and updated preconditioner.

Matrix	UPD				M			
	$nnz(L_\alpha \wedge L_R)$	$time_t$	$\|\|r\|\|_2$	itn	$nnz(L_\alpha)$	$time_t$	$\|\|r\|\|_2$	itn
BAXTER	58908	42.1	74.99	2115	31467	**39.6**	74.99	**2114**
DBIR1	40462	**1.7**	88.44	**294**	21658	107.2	88.44	22163
DBIR2	40236	**2.8**	87.57	**477**	21330	16.7	87.57	3434
NSCT1	49511	**1.2**	93.80	**261**	26610	2.0	93.80	554
NSCT2	49696	**3.0**	88.59	**701**	26693	25.1	88.59	6939
BEAFLW	73211	**6.1**	4.53	**4074**	72712	9.6	4.53	6504
PD_RHS	8860	**0.1**	34.62	**186**	4489	0.2	34.62	830
162BIT	7276	**0.2**	0.62	**389**	3800	0.2	0.62	397
176BIT	14916	**0.3**	0.80	**383**	7766	0.3	0.80	394
192BIT	27566	**0.6**	1.28	**360**	14473	0.6	1.28	373
208BIT	48552	**1.0**	1.62	**321**	25361	1.0	1.62	336
WHEEL_601	1808409	**2.9**	497.51	**32**	1084804	3.8	497.51	45
12MONTH1	24991	**24.3**	679.32	**154**	12520	30.9	679.32	182
ND_ACTORS	260367	**133.4**	301.65	**6491**	132544	140.5	301.65	7188
IMDB	6212233	113.7	497.65	1442	317616	**112.1**	497.65	**1427**
MARAGAL_6	21728	5.8	93.98	**1637**	11584	**5.6**	93.98	1771
MARAGAL_7	57532	5.2	133.13	496	31007	**4.5**	133.13	**492**
MARAGAL_8	78965	141.6	238.90	15724	45872	**129.5**	238.90	**15683**
MRI1	131131	16.4	26.74	2616	65595	**13.8**	26.74	**2537**
MRI2	150829	11.3	141.26	1692	87589	**9.7**	141.26	**1583**
TOMOGRAPHIC1	104753	**10.7**	42.18	**1309**	58845	13.1	42.18	1983

Table 2: Results for LSMR with the non-updated (M) and with the updated (UPD) preconditioners.

and number of iterations are emphasized with bold type.

The first block of matrices were not cleaned because deleting null columns and rows was not necessary. For these matrices the UPD preconditioner was able to reduce the time spent with the IC preconditioner considerably. Specially significant are the cases of the DBIR1 and NSCT2 matrices. For the second block we can observe that the UPD method is also competitive, although the improvement with respect to the IC factorization is not so big as in the previous block of matrices. In the last block, the results were not so clear, and there were cases for which the UPD preconditioner performed better, and others where it was observed the opposite.

We recall that the preconditioners used were quite sparse, that is very important for solving much larger problems.

In conclusion, the results show that the updated preconditioner method is competitive and robust for solving rank deficient least squares problems. The number of iterations and time spent was the best, or close to it, for all the problems tested.

5. Conclusions

We have presented a method for preconditioning rank deficient least squares problems that can be viewed as an update preconditioning technique for the regularized normal equations. With the regularization step a factorized preconditioner can be computed without breakdown, and with the update the preconditioner approximates better the original problem. From the numerical results conducted it has been observed that the proposed preconditioner is competitive in terms of solution time and number of iterations spent for most of the tested problems. Furthermore, a fixed value $\alpha = 1$ was large enough to avoid breakdown during the preconditioner computation and thus, recomputing the preconditioner for different values of α was not necessary. In this sense, the proposed method simplifies the choice of the Tikhonov regularization parameter. Also, with this choice we were able to compute very sparse preconditioners.

Thus, we think that the preconditioner proposed can be successfully applied to accelerate the convergence rate of the LSMR method.

References

[1] A. Bjorck, Numerical Methods for Least Squares Problems, Siam Philadelphia, 1996.

[2] C. C. Paige, M. A. Saunders, LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares, ACM Trans. Math. Softw. 8 (1) (1982) 43–71, ISSN 0098-3500, doi:10.1145/355984.355989, URL http://doi.acm.org/10.1145/355984.355989.

[3] C. C. Paige, M. A. Saunders, Algorithm 583: LSQR: Sparse Linear Equations and Least Squares Problems, ACM Trans. Math. Softw. 8 (2) (1982) 195–209, ISSN 0098-3500, doi:10.1145/355993.356000, URL http://doi.acm.org/10.1145/355993.356000.

[4] G. Golub, W. Kahan, Calculating the Singular Values and Pseudo-Inverse of a Matrix, Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis 2 (2) (1965) 205–224, doi:10.1137/0702016, URL https://doi.org/10.1137/0702016.

[5] D. C.-L. Fong, M. Saunders, LSMR: An Iterative Algorithm for Sparse Least-Squares Problems, SIAM J. Sci. Comput. 33 (5) (2011) 2950–2971, ISSN 1064-8275.

[6] J. Scott, On Using Cholesky-based Factorizations for Solving Rank-deficient Sparse Linear Least-squares Problems., SIAM J. Sci. Comput. 39 (4) (2017) C319–C339,.

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

[8] N. Li, Y. Saad, MIQR: A Multilevel Incomplete QR Preconditioner for Large Sparse Least-Squares Problems, SIAM Journal on Matrix Analysis and Applications 28 (2) (2006) 524–550, doi:10.1137/050633032, URL https://doi.org/10.1137/050633032.

[9] M. Benzi, M. Tůma, A robust incomplete factorization preconditioner for positive definite matrices, Numerical Linear Algebra with Applications 10 (5-6) (2003) 385–400, ISSN 1099-1506, doi:10.1002/nla.320, URL http://dx.doi.org/10.1002/nla.320.

[10] K. Hayami, J-F. Yin, T. Ito, GMRES Methods for Least Squares Problems., SIAM J. Matrix Anal. Appl., 31 (5) (2010) 2400–2430.

[11] R. Bru, J. Marín, J. Mas, M. Tuma, Preconditioned Iterative Methods for Solving Linear Least Squares Problems, SIAM J. Scientific Computing 36 (4).

[12] N. Gould, J. Scott, The State-of-the-Art of Preconditioners for Sparse Linear Least-Squares Problems, ACM Trans. Math. Softw. 43 (4) (2017) 36:1–36:35, ISSN 0098-3500.

[13] J. Marín, J. Mas, D. Guerrero, K. Hayami, Updating preconditioners for modified least squares problems, Num. Alg. (2017) 1–18, ISSN 1572-9265, doi:10.1007/s11075-017-0315-z.

[14] J. Cerdán, J. Marín, J. Mas, Low-rank updates of balanced incomplete factorization preconditioners, Numerical Algorithms 74 (2) (2017) 337–370.

[15] T. A. Davis, Y. Hu, The University of Florida Sparse Matrix Collection, ACM Trans. Math. Softw. 38 (1) (2011) 1:1–1:25, ISSN 0098-3500.

[16] A. Pothen, C.-J. Fan, Computing the Block Triangular Form of a Sparse Matrix, ACM Trans. Math. Softw. 16 (4) (1990) 303–324, ISSN 0098-3500, doi:10.1145/98267.98287, URL http://doi.acm.org/10.1145/98267.98287.

[17] LSMR SOFTWARE FOR LINEAR SYSTEMS AND LEAST SQUARES, URL http://web.stanford.edu/group/SOL/software/lsmr/, 2010.

[18] S. R. Arridge, M. M. Betcke, L. Harhanen, Iterated preconditioned LSQR method for inverse problems on unstructured grids, Inverse Problems 30 (7) (2014) 075009, URL http://stacks.iop.org/0266-5611/30/i=7/a=075009.