SIAM J. MATRIX ANAL. APPL.
Vol. 31, No. 2, pp. 674–693




# USING PERTURBED $QR$ FACTORIZATIONS TO SOLVE LINEAR LEAST-SQUARES PROBLEMS*

HAIM AVRON†, ESMOND NG‡, AND SIVAN TOLEDO†

**Abstract.** We propose and analyze a new tool to help solve sparse linear least-squares problems $\min_x \|Ax - b\|_2$. Our method is based on a sparse $QR$ factorization of a low-rank perturbation $\hat{A}$ of $A$. More precisely, we show that the $R$ factor of $\hat{A}$ is an effective preconditioner for the least-squares problem $\min_x \|Ax - b\|_2$, when solved using LSQR. We propose applications for the new technique. When $A$ is rank deficient, we can add rows to ensure that the preconditioner is well conditioned without column pivoting. When $A$ is sparse except for a few dense rows, we can drop these dense rows from $A$ to obtain $\hat{A}$. Another application is solving an updated or downdated problem. If $R$ is a good preconditioner for the original problem $A$, it is a good preconditioner for the updated/downdated problem $\hat{A}$. We can also solve what-if scenarios, where we want to find the solution if a column of the original matrix is changed/removed. We present a spectral theory that analyzes the generalized spectrum of the pencil $(A^*A, R^*R)$ and analyze the applications.

**Key words.** preconditioning sparse QR, iterative linear least-squares solvers

**AMS subject classifications.** 65F10, 65F20, 65F22, 65F50, 65F15

**DOI.** 10.1137/070698725

## 1. Introduction.

This paper shows that the $R$ factor from the $QR$ factorization of a perturbation $\hat{A}$ of a matrix $A$ is an effective least-squares preconditioner for $A$. More specifically, we show that the $R$ factor of the perturbation is an effective preconditioner if the perturbation can be expressed by adding or dropping a few rows from $A$ or if it can be expressed by replacing a few columns.

If $A$ is rank deficient or highly ill conditioned, the $R$ factor of a perturbation $\hat{A}$ is still an effective preconditioner if $\hat{A}$ is well conditioned. Such an $R$ factor can be used in LSQR (an iterative least-squares solver [29]) to efficiently and reliably solve a regularization of the least-squares problem. We present an algorithm for adding rows with a single nonzero to $A$ to improve its conditioning; it attempts to add as few rows as possible.

We also show that if an arbitrary preconditioner $M$ is effective for $\hat{A}^*\hat{A}$ (where $\hat{A}^*$ is the adjoint of $\hat{A}$), in the sense that the generalized condition number of $(\hat{A}^*\hat{A}, M)$ is small, then $M$ is also an effective preconditioner for $A^*A$. This shows that we do not necessarily need the $R$ factor of the perturbation $\hat{A}$; we can use $M$ as a preconditioner instead.

This paper provides a comprehensive spectral analysis of the generalized spectrum of matrix pencils that arise from row and column perturbations. The analysis shows

*Received by the editors July 30, 2007; accepted for publication (in revised form) by D. P. O'Leary September 8, 2008; published electronically June 17, 2009. This research was supported by an IBM Faculty Partnership Award, by grant 848/04 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), by grant 2002261 from the United States–Israel Binational Science Foundation, and by the Director, Office of Advanced Scientific Computing Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098.

http://www.siam.org/journals/simax/31-2/69872.html

†Blavatnik School of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel (haima@tau.ac.il, stoledo@tau.ac.il).

‡Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720 (EGNg@lbl.gov).

that if the number of rows/columns that are added, dropped, or replaced is small, then most of the generalized eigenvalues are 1 (or lie in some interval when $R$ is not an exact factor). We bound the number of *runaway* eigenvalues, which are those that are not 1 (or outside the interval), which guarantees rapid convergence of LSQR.

These results generalize a simple observation. Let $A$ be a given matrix, and let $\hat{A} = \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$. Then

$$\begin{aligned} \left(\hat{A}^*\hat{A}\right)^{-1} A^*A &= \left(A^*A + B^*B\right)^{-1} A^*A \\ &= \left(A^*A + B^*B\right)^{-1} \left(A^*A + B^*B - B^*B\right) \\ &= I - \left(A^*A + B^*B\right)^{-1} B^*B. \end{aligned} \tag{1.1}$$

The rank of the second term on the last line is at most the rank of $B$, so if $B$ has low rank, then $(\hat{A}^*\hat{A})^{-1}A^*A$ is a low-rank perturbation of the identity. A symmetric rank-$k$ perturbation of the identity has at most $k$ nonunit eigenvalues, which in exact arithmetic guarantees convergence in $k$ iterations in several Krylov-subspace iterations. Therefore, the Cholesky factor of $\hat{A}^*\hat{A}$ (which is also the $R$ factor of $\hat{A}$) is a good least-squares preconditioner for $A$. The same analysis extends to the case where we drop rows of $A$. This idea has been used by practitioners [19].

We generalize this result in additional ways: to the case where $\hat{A}$ is singular, to column exchanges, and to preconditioners for $\hat{A}$ rather than its $R$ factor. We also bound the size of the nonunit eigenvalues, which is important when $A$ is rank deficient.

The rest of this paper presents relevant background, our spectral analysis of perturbed factorizations, an algorithm for choosing the perturbations, and numerical results.

## 2. Background.

**2.1. LSQR, an iterative Krylov-subspace least-squares solver.** LSQR is a Krylov-subspace iterative method for solving the least-squares problem $\min_x \|Ax - b\|_2$. The method was developed by Paige and Saunders in 1982 [29].

The algorithm is based on the bidiagonalization procedure due to Golub and Kahan [21]. A sequence of approximations $\{x_k\}$ is generated such that the residual $\|Ax_k - b\|_2$ decreases monotonically. The sequence $\{x_k\}$ is, analytically, identical to the sequence generated by the conjugate gradients algorithm [24, 13] applied to $A^*A$. Therefore, the convergence theory of conjugate gradients applied to $A^*A$ applies directly to the behavior of LSQR. In particular, the convergence of LSQR is governed by the distribution of the eigenvalues of $A^*A$ (and can be bounded using its condition number). Another useful observation, which we will use extensively, is that if the matrix $A^*A$ has $l$ distinct eigenvalues, then LSQR will converge in at most $l$ iterations (this is a simple consequence of the minimization property of conjugate gradients).

The relationship between the condition number and the convergence of LSQR and the relationship between the number of distinct eigenvalues and the convergence of LSQR are essentially a special case of a result given by [28, Theorem 2.3] (Ng attributes the result to Van der Vorst). This result analyzes the convergence of conjugate gradients when all but $k + r$ eigenvalues lie outside a given interval. The result can also be adapted to singular matrices and LSQR, and it is used in section 5.

In this paper we use the preconditioned version of LSQR. Given an easy-to-invert preconditioner $R$, we have

$$\min_x \|Ax - b\|_2 = \min_x \|AR^{-1}Rx - b\|_2.$$

This allows us to solve $\min_x \|Ax-b\|_2$ in two phases. We first solve $\min_y \|AR^{-1}y-b\|_2$ and then solve $Rx = y$. The first phase is solved by LSQR. The convergence is now governed by the spectrum (set of eigenvalues) of $R^{-*}A^*AR^{-1}$, which is hopefully more clustered than the spectrum of $A^*A$. The spectrum of $R^{-*}A^*AR^{-1}$ is identical to the set of generalized eigenvalues $A^*Ax = \lambda R^*Rx$. We analyze these generalized eigenvalues.

**2.2. Sparse $QR$ factorizations.** In some of the applications that we describe below, the preconditioner is the $R$ factor from a $QR$ factorization of a perturbation of $A$.

The main approach to exploiting the sparsity of $A$ in a $QR$ factorization is to attempt to minimize the fill in the $R$ factor. Since the $R$ factor of $A$ is also the Cholesky factor of $A^*A$, we can use an algorithm that reduces fill in sparse Cholesky by symmetrically permuting the rows and columns of $A^*A$ [18]. Such a permutation is equivalent to a column permutation of $A$. Many algorithms can compute such a permutation without ever computing $A^*A$ or its sparsity pattern [15, 9].

When $A$ is well conditioned, it is possible to solve the least-squares problem $\min_x \|Ax-b\|_2$ using the $QR$ factorization of $A$. When $A$ is ill conditioned, it may be useful to *regularize* the equation by truncating singular values that are too small (see subsection 2.7.2 in [8]). A cheaper but effective regularization method approximates the truncated solution using a *rank revealing* $QR$ factorization of $A$ [10, 11].

Designing a sparse rank revealing $QR$ factorization is a challenging task. There are basically two techniques to compute a rank revealing $QR$ factorization. The first method, which is guaranteed to generate a rank revealing factorization, is to find a regular $QR$ factorization and refine it to a rank revealing factorization [10]. In the sparse setting the correction phase can be expensive and can produce considerable fill. We can also find a rank revealing $QR$ factorization using column pivoting [20]. This method can fail to produce a rank revealing factorization, but it usually does [17]. When $A$ is sparse, extensive column pivoting destroys the fill, reducing preordering and hence increasing fill. Column pivoting also requires more complex data structures and reduces the value of the symbolic analysis phase of the factorization.

Sparse rank revealing $QR$ factorizations do use column pivoting, usually with heuristics to restrict pivot selection (to avoid catastrophic fill). The heuristic nature of the pivot selection has a price: the ability of these factorizations to reveal rank is reduced compared to strict pivoting [12, 30]. Some algorithms [5, 2] address this problem by adding a correction phase at the end. The restricted pivoting in the first phase is aimed at reducing the amount of work that is needed in the second phase. We use this correction idea in one of our algorithms.

A sparse $QR$ algorithm can be organized in three ways. The method of George and Heath [18] rotates rows of $A$ into $R$ using Givens rotations. The multifrontal method [26] uses Householder reflections, and so does the left-looking method [14]. It is not possible to incorporate column pivoting into methods that are based on rotating rows into $R$, because there is no way to estimate the effect of pivoting on a particular column. Consequently, column-pivoting $QR$ factorizations are column-oriented and not row-oriented, in which case Householder reflections are usually used rather than Givens rotations.

**3. Preliminaries.** In this section we give some basic definitions in order to establish terminology and notation. These definitions are not new. We also restate known theorems that we will use extensively in our theoretical analysis.

DEFINITION 3.1. *Let $S$ and $T$ be $n$-by-$n$ complex matrices. We say that a scalar $\lambda$ is a* finite generalized eigenvalue *of the matrix pencil (pair) $(S,T)$ if there is a vector $v \neq 0$ such that*

$$Sv = \lambda Tv$$

*and $Tv \neq 0$. We say that $\infty$ is an* infinite generalized eigenvalue *of $(S,T)$ if there exists a vector $v \neq 0$ such that $Tv = 0$ but $Sv \neq 0$. Note that $\infty$ is an eigenvalue of $(S,T)$ if and only if $0$ is an eigenvalue of $(T,S)$. The finite and infinite eigenvalues of a pencil are* determined eigenvalues *(the eigenvector uniquely determines the eigenvalue). If both $Sv = Tv = 0$ for a vector $v \neq 0$, we say that $v$ is an* indeterminate eigenvector*, because $Sv = \lambda Tv$ for any scalar $\lambda$.*

Throughout the paper eigenvalues are ordered from smallest to largest. We will denote the $k$th eigenvalue of $S$ by $\lambda_k(S)$ and the $k$th determined generalized eigenvalue of $(S,T)$ by $\lambda_k(S,T)$. Therefore, $\lambda_1(S) \leq \cdots \leq \lambda_l(S)$ and $\lambda_1(S,T) \leq \cdots \leq \lambda_d(S,T)$, where $l$ is the number of eigenvalues that $S$ has and $d$ is the number of determined eigenvalues that $(S,T)$ has.

The solution of the least-squares equation $\min_x \|Ax - b\|_2$ is also the solution of the equation $A^*Ax = A^*x$. Matrix $A^*A$ is Hermitian positive semidefinite. The LSQR method is actually a Krylov-space method on $A^*A$, and a preconditioner for the method is Hermitian positive semidefinite too. Therefore, the matrix pencils that we will consider in this paper are *Hermitian positive semidefinite* (H/PSD) pairs.

DEFINITION 3.2. *A pencil $(S,T)$ is* Hermitian positive semidefinite *(H/PSD) if $S$ is Hermitian, $T$ is Hermitian positive semidefinite, and* $\text{null}(T) \subseteq \text{null}(S)$.

The generalized eigenvalue problem on H/PSD pencils is, mathematically, a generalization of the Hermitian eigenvalue problem. In fact, the generalized eigenvalues of an H/PSD can be shown to be the eigenvalues of an equivalent Hermitian matrix. The proof appears in [1]. Based on this observation it is easy to show that other eigenvalue properties of Hermitian matrices have an analogy for H/PSD pencils. For example, an H/PSD pencil $(S,T)$ has exactly $\text{rank}(T)$ determined eigenvalues (counting multiplicity), all of them finite and real.

A useful tool for analyzing the spectrum of a Hermitian matrix is the *Courant–Fischer Minimax Theorem* [22].

THEOREM 3.3 (Courant–Fischer Minimax Theorem). *Suppose that $S \in \mathbb{C}^{n\times n}$ is a Hermitian matrix; then*

$$\lambda_k(S) = \min_{\dim(U)=k} \max_{\substack{x\in U \\ x\neq 0}} \frac{x^*Sx}{x^*x}$$

*and*

$$\lambda_k(S) = \max_{\dim(V)=n-k+1} \min_{\substack{x\in V \\ x\neq 0}} \frac{x^*Sx}{x^*x}.$$

As discussed above, the generalized eigenvalue problem on H/PSD pencils is a generalization of the eigenvalue problem on Hermitian matrices. Therefore, there is a natural generalization of Theorem 3.3 to H/PSD pencils, which we refer to as the *Generalized Courant–Fischer Minimax Theorem.* We now state the theorem. We do not include a proof of this theorem, since it is a simple generalization of Theorem 3.3, and instead refer the reader to [1] for a complete proof.

THEOREM 3.4 (Generalized Courant–Fischer Minimax Theorem). *Suppose that* $S \in \mathbb{C}^{n\times n}$ *is a Hermitian matrix and that* $T \in \mathbb{C}^{n\times n}$ *is a Hermitian positive semidefinite matrix such that* $\text{null}(T) \subseteq \text{null}(S)$. *For* $1 \le k \le \text{rank}(T)$ *we have*

$$\lambda_k(S,T) = \min_{\substack{\dim(U)=k \\ U\perp\text{null}(T)}} \max_{x\in U} \frac{x^*Sx}{x^*Tx}$$

*and*

$$\lambda_k(S,T) = \max_{\substack{\dim(V)=\text{rank}(T)-k+1 \\ V\perp\text{null}(T)}} \min_{x\in V} \frac{x^*Sx}{x^*Tx}.$$

**4. Spectral theory.** The generalized spectrum of $(A^*A, A^*A)$ is very simple: the pencil has $\text{rank}(A)$ eigenvalues that are 1 and the rest are indeterminate. This section characterizes the structure of spectra of perturbed pencils, $(A^*A, A^*A+B^*B-C^*C)$ and $(A^*A, \tilde{A}^*\tilde{A})$, where $A = \begin{bmatrix} D & E \end{bmatrix}$ and $\tilde{A} = \begin{bmatrix} D & F \end{bmatrix}$.

These perturbations of $A^*A$ shift some of the eigenvalues of $(A^*A, A^*A)$. We call the eigenvalues that moved away from 1 *runaway* eigenvalues. This section analyzes these runaway eigenvalues, which govern the convergence of LSQR when a factorization or an approximation of the perturbed matrix is used as a preconditioner.

To keep the notation simple, we define the symmetric product $A^*A$, where $A$ is an $m$-by-$n$ matrix, to be the $n$-by-$n$ zero matrix when $m = 0$.

**4.1. Counting runaway eigenvalues.** We begin by bounding the number of runaway eigenvalues. We show that when $B$, $C$, $E$, and $F$ have low rank, the generalized eigenvalue 1 has high multiplicity in these pencils. We also bound the multiplicity of zero and indeterminate eigenvalues. The first result that we present bounds the number of runaways (and other aspects of the structure of the spectrum) when we add and/or subtract a symmetric product from a matrix. The result can be generalized to also characterize indeterminate and infinite eigenvalues, but we omit this analysis since it is not relevant to our applications.

THEOREM 4.1. *Let* $A \in \mathbb{C}^{m\times n}$, *and let* $B \in \mathbb{C}^{k\times n}$ *and* $C \in \mathbb{C}^{r\times n}$ *for some* $1 \le k + r < n$. *The following claims hold:*

1. *In the pencil* $(A^*A, A^*A+B^*B-C^*C)$, *at most* $k+r$ *generalized determined eigenvalues may be different from* 1 *(counting multiplicities).*
2. *If* 1 *is not a generalized eigenvalue of the pencil* $(B^*B, C^*C)$ *and* $A^*A + B^*B - C^*C$ *is full rank, then the multiplicity of the zero eigenvalue is exactly* $\dim\text{null}(A)$.

*Proof.* We prove most of the claims by showing that if $v$ is an eigenvector of the pencil $(A^*A, A^*A + B^*B - C^*C)$ corresponding to the eigenvalue $\lambda$, then the relationship of $v$ to the null spaces of $A$ and the relationship of $B^*Bv$ to $C^*Cv$ determine $\lambda$ in the following way:

| | $v \in \text{null}(A)$ | $v \notin \text{null}(A)$ |
|---|---|---|
| $B^*Bv = C^*Cv$ | indeterminate | $\lambda = 1$ |
| $B^*Bv \ne C^*Cv$ | $\lambda = 0$ | $\lambda \ne 0$ and $\lambda \ne 1$ |

If $v \in \text{null}(A)$ and $B^*Bv = C^*Cv$, then clearly both $A^*Av = 0$ and $(A^*A + B^*B - C^*C)v = 0$, so $v$ is an indeterminate eigenvector of $(A^*A, A^*A+B^*B-C^*C)$.

Let $v \notin \text{null}(A)$ be a vector such that $B^*Bv = C^*Cv$. Therefore,

$$(A^*A + B^*B - C^*C)\, v = A^*Av \ne 0,$$

so $v$ must be a finite generalized eigenvector of $(A^*A, A^*A + B^*B - C^*C)$ that corresponds to the eigenvalue 1.

If $v \in \text{null}(A)$ and $B^*Bv \neq C^*Cv$, then $A^*Av = 0$ and $(A^*A + B^*B - C^*C)v = A^*Av + B^*Bv - C^*Cv = B^*Bv - C^*Cv \neq 0$, so $v$ is an eigenvector corresponding to 0.

If $v \notin \text{null}(A)$ and $B^*Bv \neq C^*Cv$, then $\lambda$ can be neither 0 nor 1. It cannot be 0 because $A^*Av \neq 0$. It cannot be 1 because that would imply $B^*Bv - C^*Cv = 0$, which is a contradiction to the assumption that $B^*Bv \neq C^*Cv$.

To establish claim 1 notice that if $v \in \text{null}(B) \cap \text{null}(C)$, then clearly $B^*Bv = C^*Cv$. So, if $v$ is a determined generalized eigenvector corresponding to an eigenvalue different from 1, then $v \notin \text{null}(B) \cap \text{null}(C)$. Therefore, the dimension of the space spanned by these vectors is bounded by $\dim((\text{null}(B) \cap \text{null}(C))^{\perp}) \leq k + r$, which bounds the number of such eigenvalues.

We now turn our attention to claim 2. Assume that $A^*A + B^*B - C^*C$ is full rank and 1 is not a generalized eigenvalue of the pencil $(B^*B, C^*C)$. The multiplicity of the eigenvalue 0 follows from the fact that every $0 \neq v \in \text{null}(A)$ satisfies $A^*Av = 0$ and $(A^*A + B^*B - C^*C)v \neq 0$ (because $A^*A + B^*B - C^*C$ has full rank). Therefore, $v$ is indeed a generalized eigenvector. The converse is true from the table and the fact that $B^*Bv \neq C^*Cv$ for every vector $v$. ∎

The second result of this section characterizes the generalized spectra of symmetric products that are formed by modifying a set of columns in a given matrix $A$. We denote the columns of $A$ that are not modified in the factorization by $D$, the columns that are to be modified by $E$, and the new value in those columns by $F$.

THEOREM 4.2. *Let $D \in \mathbb{C}^{m\times n}$, and let $E \in \mathbb{C}^{m\times k}$ and $F \in \mathbb{C}^{m\times k}$ for some $1 \leq k < n$. Let*

$$A = \begin{bmatrix} D & E \end{bmatrix} \in \mathbb{C}^{m\times(n+k)},$$

*and let*

$$\tilde{A} = \begin{bmatrix} D & F \end{bmatrix} \in \mathbb{C}^{m\times(n+k)}.$$

*In the pencil $(A^*A, \tilde{A}^*\tilde{A})$, at least $n - k$ generalized finite eigenvalues are* 1.

*Proof.* Expanding $A^*A$ and $\tilde{A}^*\tilde{A}$, we obtain

$$A^*A = \begin{bmatrix} D^* \\ E^* \end{bmatrix} \begin{bmatrix} D & E \end{bmatrix} = \begin{bmatrix} D^*D & D^*E \\ E^*D & E^*E \end{bmatrix}$$

and

$$\tilde{A}^*\tilde{A} = \begin{bmatrix} D^* \\ F^* \end{bmatrix} \begin{bmatrix} D & F \end{bmatrix} = \begin{bmatrix} D^*D & D^*F \\ F^*D & F^*F \end{bmatrix}.$$

Let $S$ be the vector space in $\mathbb{C}^{n+k}$ defined by

$$S = \left\{ \begin{bmatrix} v \\ 0 \end{bmatrix} : v \in \mathbb{C}^n \text{ such that } E^*Dv = F^*Dv \right\}.$$

Clearly, $\dim S = \dim \text{null}(E^*D - F^*D) = n - \text{rank}(E^*D - F^*D)$. The matrix $E^*D - F^*D$ has $k$ rows, so $\text{rank}(E^*D - F^*D) \leq k$, which implies $\dim S \geq n - k$.

Let $v$ be a vector such that $E^*Dv = F^*Dv$. The vector $\left[\begin{smallmatrix} v \\ 0 \end{smallmatrix}\right]$ is a generalized eigenvector of $(A^*A, \tilde{A}^*\tilde{A})$ corresponding to the eigenvalue 1, because

$$A^*A \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} D^*D & D^*E \\ E^*D & E^*E \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} D^*Dv \\ E^*Dv \end{bmatrix}$$
$$= \begin{bmatrix} D^*Dv \\ F^*Dv \end{bmatrix} = \begin{bmatrix} D^*D & D^*F \\ F^*D & F^*F \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \tilde{A}^*\tilde{A} \begin{bmatrix} v \\ 0 \end{bmatrix}.$$

Since $S$ is a subset of the generalized eigenspace of $(A^*A, \tilde{A}^*\tilde{A})$ corresponding to the eigenvalue 1, the multiplicity of 1 as a generalized eigenvalue is at least $\dim S \geq n - k$. □

**4.2. Runaways in a preconditioned system.** We now show that if a preconditioner $M$ is effective for a matrix $A^*A$, then it is also effective for the perturbed matrices $A^*A + B^*B - C^*C$ and $\tilde{A}^*\tilde{A}$. If the rank of the matrices $B$, $C$, $E$, and $F$ is low, then most of the generalized eigenvalues of the perturbed preconditioned system will be bounded by the extreme generalized eigenvalues of the unperturbed preconditioned system. In other words, the number of runaways is still guaranteed to be small, but the nonrunaways are not necessarily at 1: they can move about the interval whose size determines the condition number of the original preconditioned system. Ng in [28, Theorem 2.3] shows that this spectral characterization guarantees rapid convergence; in exact arithmetic, after an iteration for each row in $B$ and $C$, the convergence rate bound is governed by the unperturbed condition number (after two iterations for every column exchanged for column perturbations).

THEOREM 4.3. *Let $A \in \mathbb{C}^{m\times n}$, and let $B \in \mathbb{C}^{k\times n}$ and $C \in \mathbb{C}^{r\times n}$ for some $1 \leq k + r < n$. Let $M \in \mathbb{C}^{n\times n}$ be a Hermitian positive semidefinite matrix. Suppose that* $\mathrm{null}(M) \subseteq \mathrm{null}(A^*A)$, $\mathrm{null}(M) \subseteq \mathrm{null}(B^*B)$, *and* $\mathrm{null}(M) \subseteq \mathrm{null}(C^*C)$. *If*

$$\alpha \leq \lambda_1(A^*A, M) \leq \lambda_{\mathrm{rank}(M)}(A^*A, M) \leq \beta,$$

*then*

$$\alpha \leq \lambda_{r+1}(A^*A + B^*B - C^*C, M) \leq \lambda_{\mathrm{rank}(M)-k}(A^*A + B^*B - C^*C, M) \leq \beta.$$

*Proof.* Denote $t = \mathrm{rank}(M)$. We prove the lower bound using the second equality of Theorem 3.4. Let $p = \mathrm{rank}(C)$; we have

$$\lambda_{p+1}(A^*A + B^*B - C^*C, M) = \max_{\substack{\dim(U)=t-p \\ U\perp\mathrm{null}(M)}} \min_{x\in U} \frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx}.$$

We prove the bound by showing that for one specific $U$, the ratio for any $x \in U$ is at least $\alpha$. This implies that the minimum ratio in $U$ is at least $\alpha$, and that the maximum over all admissible subspaces $U$ is also at least $\alpha$.

Let $U = \mathrm{null}(C^*C) \cap \mathrm{range}(M)$. Because $M$ is Hermitian positive semidefinite, $\mathrm{null}(M) \perp \mathrm{range}(M)$. This implies that $U \perp \mathrm{null}(M)$. Since $\mathrm{null}(M) \subseteq \mathrm{null}(C^*C)$, we have $\dim(U) = t - p$ (here we use $\mathrm{range}(C^*C) \subseteq \mathrm{range}(M)$). For every $x \in U$ we

have $x^*(A^*A + B^*B - C^*C)x = x^*(A^*A + B^*B)x \geq x^*A^*Ax$, so

$$\begin{aligned}\frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx} &\geq \frac{x^*A^*Ax}{x^*Mx} \\ &\geq \min_{x \in \text{range}(M)} \frac{x^*A^*Ax}{x^*Mx} \\ &= \lambda_1(A^*A, M) \\ &\geq \alpha.\end{aligned}$$

Therefore,

$$\min_{x \in U} \frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx} \geq \alpha,$$

so $\lambda_{p+1}(A^*A + B^*B - C^*C, M) \geq \alpha$. Since $p = \text{rank}(C) \leq r$, we have shown that

$$\lambda_{r+1}(A^*A + B^*B - C^*C, M) \geq \lambda_{p+1}(A^*A + B^*B - C^*C, M) \geq \alpha.$$

For the upper bound we use a similar strategy, but with the first equality of Theorem 3.4. Let $l = \text{rank}(B)$; we have

$$\lambda_{t-l}(A^*A + B^*B - C^*C, M) = \min_{\substack{\dim(V) = t-l \\ V \perp \text{null}(M)}} \max_{x \in V} \frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx}.$$

Let $V = \text{null}(B^*B) \cap \text{range}(M)$. Since $M$ is Hermitian positive semidefinite, $V \perp \text{null}(M)$ and $\dim(V) = (n-l) - (n-t) = t - l$. For every $x \in V$ we have $x^*(A^*A + B^*B - C^*C)x = x^*(A^*A - C^*C)x \leq x^*A^*Ax$, so

$$\begin{aligned}\frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx} &\leq \frac{x^*A^*Ax}{x^*Mx} \\ &\leq \max_{x \in \text{range}(M)} \frac{x^*A^*Ax}{x^*Mx} \\ &= \lambda_t(A^*A, M) \\ &\leq \beta.\end{aligned}$$

Since

$$\max_{x \in V} \frac{x^*(A^*A + B^*B - C^*C)x}{x^*Mx} \leq \beta,$$

we have $\lambda_{t-l}(A^*A + B^*B - C^*C, M) \leq \beta$, and since $l = \text{rank}(B) \leq k$, we have shown that

$$\lambda_{t-k}(A^*A + B^*B - C^*C, M) \leq \lambda_{t-l}(A^*A + B^*B - C^*C, M) \leq \beta. \qquad \square$$

We now give the analogous theorem when columns are modified.

THEOREM 4.4. *Let $D \in \mathbb{C}^{m \times n}$, and let $E \in \mathbb{C}^{m \times k}$ and $F \in \mathbb{C}^{m \times k}$ for some $1 \leq k < n$. Let*

$$A = \begin{bmatrix} D & E \end{bmatrix} \in \mathbb{C}^{m \times (n+k)},$$

*and let*

$$\tilde{A} = \begin{bmatrix} D & F \end{bmatrix} \in \mathbb{C}^{m \times (n+k)}.$$

*Let* $M \in \mathbb{C}^{(n+k)\times(n+k)}$ *be a Hermitian positive semidefinite matrix, such that* $\mathrm{null}(M) \subseteq \mathrm{null}(A^*A)$ *and* $\mathrm{null}(M) \subseteq \mathrm{null}(\tilde{A}^*\tilde{A})$. *Suppose that*

$$\alpha \leq \lambda_1(A^*A, M) \leq \lambda_{\mathrm{rank}(M)}(A^*A, M) \leq \beta.$$

*Then we have*

$$\alpha \leq \lambda_{k+1}(\tilde{A}^*\tilde{A}, M) \leq \lambda_{\mathrm{rank}(M)-k}(\tilde{A}^*\tilde{A}, M) \leq \beta.$$

*Proof.* We denote $t = \mathrm{rank}(M)$ and $r = \mathrm{rank}(E^*D - F^*D) \leq k$ (because $E^*D - F^*D$ has $k$ rows). We prove both sides by applying Theorem 3.4. We define the linear subspace of $\mathbb{C}^{n+k}$:

$$U = \left\{ \begin{bmatrix} v \\ 0 \end{bmatrix} \,:\, v \in \mathbb{C}^n \text{ and } E^*Dv = F^*Dv \right\} \cap \mathrm{range}(M).$$

Clearly, $U$ is a linear space and $U \perp \mathrm{null}(M)$. For any $\left[\begin{smallmatrix} v \\ 0 \end{smallmatrix}\right] \in \mathrm{null}(M)$, the vector $v \in \mathbb{C}^n$ satisfies $E^*Dv = F^*Dv = 0$, because $\mathrm{null}(M) \subseteq \mathrm{null}(A^*A)$ and $\mathrm{null}(M) \subseteq \mathrm{null}(\tilde{A}^*\tilde{A})$. This implies that the set of $v$'s for which $v \in \mathrm{null}(E^*D - F^*D)$ contains the set of $v$'s for which $\left[\begin{smallmatrix} v \\ 0 \end{smallmatrix}\right] \in \mathrm{null}(M)$. This allows us to determine the dimension of $U$:

$$\begin{aligned} \dim(U) &= \dim \mathrm{null}(E^*D - F^*D) - \dim\left(\left\{ v \in \mathbb{C}^n \,:\, \begin{bmatrix} v \\ 0 \end{bmatrix} \in \mathrm{null}(M) \right\}\right) \\ &= (n - r) - (n - t) \\ &= t - r. \end{aligned}$$

It is easy to see that for every $x \in U$ we have $A^*Ax = \tilde{A}^*\tilde{A}x$, so

$$\begin{aligned} \frac{x^*\tilde{A}^*\tilde{A}x}{x^*Mx} &= \frac{x^*A^*Ax}{x^*Mx} \\ &\geq \min_{x \in \mathrm{range}(M)} \frac{x^*A^*Ax}{x^*Mx} \\ &= \lambda_1(A^*A, M) \\ &\geq \alpha. \end{aligned}$$

Since, by the second equality of Theorem 3.4,

$$\lambda_{r+1}(\tilde{A}^*\tilde{A}, M) = \max_{\substack{\dim U = t-r \\ U \perp \mathrm{null}(M)}} \min_{x \in U} \frac{x^*\tilde{A}^*\tilde{A}x}{x^*Mx},$$

we conclude that $\lambda_{k+1}(\tilde{A}^*\tilde{A}, M) \geq \lambda_{r+1}(\tilde{A}^*\tilde{A}, M) \geq \alpha$. Similarly, for every $x \in U$,

$$\begin{aligned} \frac{x^*\tilde{A}^*\tilde{A}x}{x^*Mx} &= \frac{x^*A^*Ax}{x^*Mx} \\ &\leq \max_{x \in \mathrm{range}(M)} \frac{x^*A^*Ax}{x^*Mx} \\ &= \lambda_t(A^*A, M) \\ &\leq \beta. \end{aligned}$$

Since, by the first equality of Theorem 3.4,

$$\lambda_{t-r}(\tilde{A}^*\tilde{A}, M) = \min_{\substack{\dim U = t-r \\ U \perp \mathrm{null}(M)}} \max_{x \in U} \frac{x^*\tilde{A}^*\tilde{A}x}{x^*Mx},$$

we conclude that $\lambda_{t-k}(\tilde{A}^*\tilde{A}, M) \leq \lambda_{t-r}(\tilde{A}^*\tilde{A}, M) \leq \beta$. ∎

**4.3. Using the simple spectrum of $A^*A$ to bound the magnitude of runaways.** In some cases it is useful to know that runaway eigenvalues are either very small or very close to 1. For example, we want to ensure that if we perturb an ill-conditioned $A^*A$ to a well-conditioned $A^*A + B^*B$, the numerical rank of $A^*A$ and of $(A^*A, A^*A + B^*B)$ are the same, up to an appropriate relaxation of the rank threshold. We need the following lemma.

LEMMA 4.5. *Suppose that $S \in \mathbb{C}^{n\times n}$ is a Hermitian matrix and that $T \in \mathbb{C}^{n\times n}$ is a Hermitian positive definite matrix. For all $1 \leq k \leq n$ we have*

$$\frac{\lambda_k(S)}{\lambda_n(T)} \leq \lambda_k(S,T) \leq \frac{\lambda_k(S)}{\lambda_1(T)}.$$

*Proof.* Let $u_1, \ldots, u_n$ be a set of orthonormal eigenvectors corresponding to $\lambda_1(S), \ldots, \lambda_n(S)$. Using subspaces $U_k = \mathrm{span}\{u_1, \ldots, u_k\}$ and $V_k = \mathrm{sp}\{u_k, \ldots, u_n\}$ in the first inequality and second inequality of Theorem 3.4, respectively, gives the two bounds. ∎

We now state and prove the main results.

THEOREM 4.6. *Let $A \in \mathbb{C}^{m\times n}$, and let $B \in \mathbb{C}^{k\times n}$ for some $1 \leq k < n$. Assume that $A^*A + B^*B$ is full rank. Denote $\alpha = ||A^*A||_2$. If there are $d$ eigenvalues of $A^*A$ that are smaller than or equal to $\epsilon\alpha$ for some $1 > \epsilon > 0$, then $d$ generalized eigenvalues of $(A^*A, A^*A + B^*B)$ are smaller than or equal to $\epsilon\kappa(A^*A + B^*B)$.*

*Proof.* We denote $S = A^*A$ and $T = A^*A + B^*B$. We first note that $\lambda_n(T) \geq \lambda_n(S) \geq \alpha$. By the lemma,

$$\begin{aligned}\lambda_k(S,T) &\leq \frac{\lambda_k(S)}{\lambda_1(T)} = \frac{\lambda_k(S)\lambda_n(T)}{\lambda_n(T)\lambda_1(T)} \\ &= \frac{\lambda_k(S)}{\lambda_n(T)}\kappa(T) \\ &\leq \frac{\lambda_k(S)}{\alpha}\kappa(T).\end{aligned}$$

For any $k$ such that $\lambda_k(S) \leq \epsilon\alpha$, we obtain the desired inequality. ∎

THEOREM 4.7. *Let $A \in \mathbb{C}^{m\times n}$, and let $B \in \mathbb{C}^{k\times n}$ for some $1 \leq k < n$. Assume that $A^*A + B^*B$ is full rank. Denote $\alpha = ||A^*A||_2$ and suppose that $||B^*B||_2 \leq \gamma\alpha$. If there are $d$ eigenvalues of $A^*A$ that are larger than or equal to $\eta\alpha$ for some $1 > \eta > 0$, then $d$ generalized eigenvalues of $(A^*A, A^*A + B^*B)$ are larger than or equal to $\eta/(1+\gamma)$.*

*Proof.* We use the same notation as in the previous proof. We have $\lambda_n(T) \leq ||A^*A||_2 + ||B^*B||_2 \leq (1+\gamma)\alpha$. Therefore,

$$\begin{aligned}\lambda_k(S,T) &\geq \frac{\lambda_k(S)}{\lambda_n(T)} \\ &\geq \frac{\lambda_k(S)}{(1+\gamma)\alpha},\end{aligned}$$

which gives the desired bound for any $k$ such that $\lambda_k(S) \geq \eta\alpha$. ∎

The theorems show that the numerical rank of the preconditioned system is the same as that of the original system, up to an appropriate relaxation of the rank threshold. Suppose that after the $d$th eigenvalue there is a big gap. That is, there are $d$ eigenvalues of $A^*A$ that are smaller than $\epsilon\alpha$, and the remaining $n-d$ are larger than $\eta\alpha$, where $\alpha$ is the largest eigenvalue of $A^*A$. The ratio between the largest eigenvalue and the $d$th smallest is at least $1/\epsilon$, and between the largest eigenvalue and the $(n-d)$th largest is at most $1/\eta$. Recall that 1 is the largest eigenvalue of $(A^*A, A^*A + B^*B)$. Therefore, the ratio between the largest eigenvalue of the pencil and the $d$ smallest is at least $\kappa^{-1}(A^*A + B^*B)/\epsilon$, and the ratio between the largest eigenvalue of $(A^*A, A^*A+B^*B)$ and the $n-d$ largest eigenvalues is at most $(1+\gamma)/\eta$. Therefore, if $B^*B$ is not too large relative to $A^*A$, and $A^*A+B^*B$ is well conditioned, then the ratios are roughly maintained.

In section 6 below we present an efficient algorithm that finds a $B$ such that $||B^*B||_2 \leq m||A^*A||_2$ and $\kappa(A^*A + B^*B) \leq \tau^2$, where $\tau \geq n+1$ is a given threshold, and a slightly more expensive algorithm that requires only $\tau \geq \sqrt{2n}$ and guarantees $||B^*B||_2 \leq ||A^*A||_2$.

**5. Applications to least-squares solvers.** This section describes applications of the theory to the solution of linear least-squares problems. We show that we can often obtain useful algorithms by combining a sparse $QR$ factorization of a modified matrix with a preconditioned iterative solver. We focus on improving the utility and efficiency of sparse $QR$ factorizations and not on the more general problem of finding effective preconditioners.

In all the applications, we compute the $R$ factor of a $QR$ factorization of a modified matrix and use it as a preconditioner in LSQR. Our spectral theory in section 4 shows that the preconditioned system has only a few runaway eigenvalues. We then can use [28, Theorem 2.3] to bound the number of iterations.

**5.1. Dropping dense rows for sparsity.** The $R$ factor of $A = QR$ is also the Cholesky factor of $A^*A$. Rows of $A$ that are fairly dense cause $A^*A$ to be fairly dense. Hence, $R$ will be dense. In the extreme case, a completely dense row in $A$ causes $A^*A$ and $R$ to be completely dense (unless there are exact cancellations, which are rare). This happens even if the other rows of $A$ all have a single nonzero.

If $A$ has a few rows that are fairly dense, we recommend that they be dropped before the $QR$ factorization starts. More precisely, these rows should be dropped even before the column ordering is computed. If we dropped $k$ dense rows, we expect LSQR to converge in at most $k+1$ iterations (see subsection 2.1).

Heath [23] proposed a different method for handling dense rows (see also [7] and [27]), in which the dominant costs are the factorization of the first $m$ rows of $A$ (the same is true in our approach), $k$ triangular solves with $R^*$, and a dense $QR$ factorization of an $(n+k)$-by-$k$ matrix. In most cases (e.g., when $R$ is denser than $A$), the asymptotic cost of each of the two methods is similar; there are also cases in which one method is cheaper than the other (in both directions).

**5.2. Updating and downdating.** Updating a least-squares problem involves adding rows to the coefficient matrix $A$ and to the right-hand-side $b$. Downdating involves dropping rows. Suppose that we factored the original coefficient matrix $A$, that updating added additional rows represented by a matrix $B$, and that downdating removed rows of $A$ that are represented by a matrix $C$. The coefficient matrix of the normal equations of the updated/downdated problem is $A^*A + B^*B - C^*C$. As long

as this coefficient matrix is full rank and the number of rows in $B$ and $C$ is small, Theorem 4.1 guarantees that the $R$ factor of $A$ is an effective preconditioner.

**5.3. Adding rows to solve numerically rank deficient problems.** We propose two methods for solving numerically rank deficient problems.

**5.3.1. Using an iterative method.** When $A$ is rank deficient, there is an entire subspace of minimizers of $\|Ax - b\|_2$. When $A$ is full rank but highly ill conditioned, there is a single minimizer, but there are many $x$'s that give almost the same residual norm. Of these minimizers or almost-minimizers, the user usually prefers a solution with a small norm.

The factorization $A = QR$ is not useful for solving rank deficient and ill-conditioned least-squares problems. The factorization is backward stable, but the computed $R$ is ill conditioned. This usually causes the solver to produce a solution $x = R^{-1}Q^*b$ with a huge norm. This also happens if we use $R$ as a preconditioner in LSQR: the iterations stop after one or two steps with a solution with a huge norm. Even after the first iteration the solution vector has a huge norm.

The singular-value decomposition (SVD) and rank revealing $QR$ factorizations can produce minimal-norm solutions, but they are difficult to compute. The SVD approach is not practical in the sparse case. The rank revealing $QR$ approach is practical [30, 5, 2, 12, 23], but sparse rank revealing $QR$ factorizations are complex, and only a few implementations are available.

The approach that we propose here is to add rows to the coefficient matrix $A$ to avoid ill-conditioning in $R$. That is, we dynamically add rows to $A$ to avoid ill-conditioning in $R$. The factor $R$ is no longer the $R$ factor of $A$ but the $R$ factor of a perturbed matrix $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$, where $B$ consists of the added rows. Section 6 outlines an algorithm for dynamically adding rows to $A$ so that the $R$ factor of the perturbed matrix will not be ill conditioned.

The well-conditioned $R$ factor of the perturbed matrix is then used as a preconditioner for LSQR. The convergence threshold of LSQR allows the user to control a trade-off between the norm of the residual and the norm of the solution. Suppose that the user wishes to find a minimizer of $\min_x \|\bar{A}x - b\|_2$, where $\bar{A}$ has the same SVD as $A$ except that the $k$ smallest singular values of $A$ are replaced by 0. When LSQR's convergence threshold is larger than $r = \sigma_{n-k}/\sigma_1$, it computes such a minimizer [29].

When the $R$ factor of $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ is used as a preconditioner, correct truncation at $\sigma_{n-k}$ depends on the preconditioned system preserving the singular gap above $\sigma_{n-k}$. This is why the results in subsection 4.3 are important: they guarantee this preservation.

In exact arithmetic, the number of rows in $B$ bounds the number of iterations in LSQR. It may be smaller if the runaway eigenvalues are clustered.

**5.3.2. Using a direct method.** If the number of rows in $B$ is *exactly* the same as the number of singular values we wish to truncate, and if $A^*A + B^*B$ is well conditioned, then a direct method can find an approximation of a small-norm minimizer. Let $A \in \mathbb{C}^{m\times n}$, and let $B \in \mathbb{C}^{k\times n}$. Let $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] = QR$ and $P = \left[\begin{array}{cc} I_{m\times m} & 0_{m\times k} \end{array}\right]$. We show that if the $k$ smallest singular values of $A$ are small enough, then $\hat{x} = R^{-1}(PQ)^*b$ is close to a minimizer of $\min_x \|\bar{A}x - b\|_2$, as defined above.

We start with a simple lemma that forms the basis of our method.

LEMMA 5.1. *Let $A \in \mathbb{C}^{m\times n}$, and let $B \in \mathbb{C}^{k\times n}$. Suppose that* $\operatorname{rank}(A) = n - k$ *and that $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ has full rank. Let $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] = QR$ be a QR factorization of $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$. All the singular values of $AR^{-1}$ are exactly* 0 *or* 1.

*Proof.* The singular values of $AR^{-1}$ are the square root of the eigenvalues of $R^{-*}A^*AR^{-1}$. The eigenvalues $R^{-*}A^*AR^{-1}$ are exactly the eigenvalues of $(A^*A, R^*R)$. It is easy to see that $R^*R = A^*A + B^*B$. If we apply claim 2 of Theorem 4.1, we conclude that the multiplicity of the 0 eigenvalue of $(A^*A, R^*R)$ is exactly $\dim \text{null}(A) = n - \text{rank}(A) = k$, and the multiplicity of the 1 eigenvalue of $(A^*A, R^*R)$ is exactly $n - \text{rank}(B) = n - k$. Therefore, $n$ eigenvalues of $(A^*A, R^*R)$, which are all the eigenvalues of $(A^*A, R^*R)$, are either 0 or 1. ∎

We now show our claim for the case that $A$ is exactly rank deficient by $k$, so $\bar{A} = A$. This is a simpler case than the case where the $k$ smallest singular values are small but not necessarily zero. In this case the vector $\hat{x} = R^{-1}(PQ)^*b$ is an exact minimizer of $\min_x \|\bar{A}x - b\|_2$.

LEMMA 5.2. *Let $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{k\times n}$, and $b \in \mathbb{C}^{m\times r}$. Suppose that* $\text{rank}(A) = n - k$ *and* $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$ *has full rank. Let* $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] = QR$ *be a QR factorization of* $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right]$. *Let* $P = \left[\begin{smallmatrix} I_{m\times m} & 0_{m\times k} \end{smallmatrix}\right]$. *The vector $\hat{x} = R^{-1}(PQ)^*b$ is a minimizer of* $\min_x \|Ax - b\|_2$.

*Proof.* We show that $\hat{y} = R\hat{x} = (PQ)^*b$ is the minimum norm solution to $\min_y \|AR^{-1}y - b\|_2$. The minimum solution norm to $\min_y \|AR^{-1}y - b\|_2$ is

$$y_{\min} = \left(AR^{-1}\right)^+ b.$$

According to Lemma 5.1, the singular values of $AR^{-1}$ are exactly 0 and 1. Therefore,

$$\left(AR^{-1}\right)^+ = \left(AR^{-1}\right)^*.$$

Notice that

$$AR^{-1} = P\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] R^{-1} = PQRR^{-1} = PQ.$$

Therefore, $y_{\min} = (PQ)^*b = \hat{y}$. ∎

We now analyze the case where $A$ is has $k$ small but possibly nonzero singular values. In this case, $\hat{x} = R^{-1}(PQ)^*b$ is not necessarily a minimizer of $\min_x \|Ax - b\|_2$ and, more importantly, not even a minimizer of $\min_x \|\bar{A}x - b\|_2$. But if $\left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] = \bar{Q}\bar{R}$, then the vector $\hat{z} = \bar{R}^{-1}(P\bar{Q})^*b$ is a minimizer of $\min_x \|\bar{A}x - b\|_2$. If the truncated singular values are small enough, then the pairs $(Q, R)$ and $(\bar{Q}, \bar{R})$ will be closely related because they are $QR$ factorizations of nearby matrices. Therefore, $\hat{x}$ and $\hat{z}$ should not be too far from each other. The next theorem shows that this is indeed the case.

THEOREM 5.3. *Let $A \in \mathbb{C}^{m\times n}$, $B \in \mathbb{C}^{k\times n}$, and $b \in \mathbb{C}^m$. Let $\bar{A}$ be the matrix with the same SVD as $A$ except that the $k$ smallest singular values are truncated to* 0. *Denote*

$$C = \left[\begin{smallmatrix} A \\ B \end{smallmatrix}\right] \text{ and } D = \left[\begin{smallmatrix} \bar{A} \\ B \end{smallmatrix}\right].$$

*Assume that $C$ and $D$ are both full rank. Let $C = QR$ be a $QR$ factorization of $C$ and $D = \bar{Q}\bar{R}$ be the $QR$ factorization of $D$. Denote*

$$\delta = \frac{\sigma_{n-k+1}(A)}{\sigma_{\min}(C)},$$

*where $\sigma_{n-k}(A)$ is the $k$th smallest singular value of $A$ and $\sigma_{\min}(C)$ is the smallest singular value of $C$. Let $P = \left[\begin{smallmatrix} I_{m\times m} & 0_{m\times k} \end{smallmatrix}\right]$. Define the solutions*

$$\hat{x} = R^{-1}(PQ)^*b$$

*and*

$$\hat{z} = \bar{R}^{-1}(P\bar{Q})^* b.$$

*Then, provided that* $\delta < 1$,

$$\frac{\|\hat{x} - \hat{z}\|_2}{\|\hat{x}\|_2} \leq \frac{\delta}{1-\delta}\left(2 + (\kappa(R)+1)\frac{\|r\|_2}{\|R\|_2\|\hat{z}\|_2}\right),$$

*where*

$$r = b - A\hat{x}.$$

Before we prove the theorem, we explain what it means. The algorithm computes $\hat{x}$ and can therefore compute $r = b - A\hat{x}$. The theorem states that if $\delta$ is small (which happens when $C$ is well conditioned and $A$ has $k$ tiny singular values), if $R$ is not ill conditioned and not too large, and if the norm of $r$ is not too large, then $\hat{x}$ is a good approximation of the minimizer $\hat{z}$ that we seek. The quantity that is hard to estimate in practice is $\delta$, which depends on the small singular values of $A$. Therefore, the method is useful mainly when we know a priori the number of small singular values of $A$.

*Proof.* Notice that $\hat{x}$ is the solution of $\min_x \|Cx - P^*b\|_2$ and that $\hat{z}$ is the solution of $\min_x \|Dx - P^*b\|_2$. Furthermore, we can write $D = C + \Delta C$, where $\|\Delta C\|_2 \leq \sigma_{n-k+1}(A)$. If we define $\epsilon = \sigma_{n-k+1}(A)/\|C\|_2$, then $\kappa(C)\epsilon = \delta < 1$ and $\|\Delta C\|_2 \leq \epsilon\|C\|_2$. We can apply a variant of result from Wedin [31] (see [25, Theorem 2.1] for the specific version that we use) and conclude that

$$\frac{\|\hat{x} - \hat{z}\|_2}{\|\hat{x}\|_2} \leq \frac{\delta}{1-\delta}\left(2 + (\kappa(R)+1)\frac{\|r\|_2}{\|R\|_2\|\hat{z}\|_2}\right). \qquad \square$$

**5.4. Solving what-if scenarios.** The theory presented in this paper allows us to efficiently solve what-if scenarios of the following type. We are given a least squares problem $\min \|Ax - b\|_2$. We already computed the minimizer using the $R$ factor of $A$ or using some preconditioners. Now we want to know how the solution would change if we fix some of its entries, without loss of generality $x_{n-k+1} = c_{n-k+1}, \ldots, x_n = c_n$, where the $c_i$'s are some constants. We denote $A = \begin{bmatrix} D & E \end{bmatrix}$, where $E$ consists of $k$ columns. To solve the what-if scenario, we need to solve $\min \|Dx_{1:n-k} - (b - Ec)\|_2$. We solve instead $\min \|\tilde{A}x - (b - Ec)\|_2$, where $\tilde{A} = \begin{bmatrix} D & 0 \end{bmatrix}$, a matrix that we obtain from $A$ by replacing the last $k$ columns by zeros. Clearly, the last $k$ entries of $x$ do not influence the norm of the residual in this system, so we can ignore them. By Theorem 4.2, for small $k$ the factor or the preconditioner of $A$ is effective for this least-squares system as well.

**6. An algorithm for perturbing to improve the conditioning.** In this section we show an algorithm that perturbs a given input matrix $A$ to improve its conditioning. The algorithm adds only rows, which all have a single nonzero. The algorithm finds the perturbation during and after a standard Householder $QR$ factorization (the technique applies to any column-oriented $QR$ algorithm). Therefore, it can be easily integrated into a sparse $QR$ factorization code; unlike rank revealing $QR$ algorithms, our algorithm does not exchange columns.

The goal of the algorithm is to build an $R$ whose condition number is below a given threshold $\tau$, with as few modifications as possible. More specifically, the goal is to find a $B \in \mathbb{C}^{k\times n}$ and upper triangular $R \in \mathbb{C}^{n\times n}$ such that

1. $A^*A + B^*B = R^*R$,
2. the Cholesky factors of $A^*A$ and $A^*A+B^*B$ are structurally the same (except for accidental cancellations),
3. $\kappa(R) \leq \tau$, and
4. $k$ is small.

We ensure that the first goal is met as follows. If, during the factorization of column $j$, the algorithm finds that it needs to add a row to $B$, it adds a row with zeros in columns 1 to $j-1$. This ensures that the first $j-1$ columns computed so far are also the factor of the newly perturbed matrix. (In fact, it always adds a row with a nonzero only in column $j$.)

By restricting the number of nonzeros in each row of $B$ to one, we automatically achieve the second goal, since $B^*B$ is diagonal.

The algorithm works in two stages. In the first stage, the matrix is perturbed during the Householder $QR$ factorization. In step $j$, we factor column $j$ and then run a condition-number estimator to detect ill-conditioning in the leading $j$-by-$j$ block of $R$. If this block is ill conditioned, we add a row to $B$, which causes only $R_{j,j}$ to change. A trivial condition estimation technique is to estimate the large singular value of $A$ using its one or infinity norm and then to estimate the smallest singular value using the smallest diagonal element in $R$. This method, however, is not always reliable. There are incremental condition estimators for triangular matrices that are efficient and more reliable [4, 6, 3, 16].

Let $c_A = ||A||_1$ be an estimation of the norm of $A$. Other norms can be used and will modify some of the values below. All the rows of $B$ will be completely zero except a single nonzero, which we set to $\pm c_A$. Each row of $B$ has a different nonzero column. It follows that $B^*B$ is diagonal to $||B||_2 = c_A$. Therefore,

$$\begin{aligned} ||R||_2 &= \sqrt{||R^*R||_2} = \sqrt{||A^*A + B^*B||_2} \\ &\leq \sqrt{||A^*A||_2 + ||B^*B||_2} \\ &\leq \sqrt{nc_A^2 + c_A^2} \\ &\leq c_A\sqrt{n+1}. \end{aligned}$$

Therefore, we add a row to $B$ whenever the incremental condition estimator suspects that $||R^{-1}||_2 > \tau/c_A\sqrt{n+1}$. If we estimate $c_A = ||A||_2$ directly (using power iteration), we need only to ensure that $||R^{-1}||_2 > \tau/c_A\sqrt{2}$, so we can use fewer perturbations.

Condition estimators can fail to detect ill-conditioning. For example, if we estimate $\|R^{-1}\|_2 \approx 1/\min_j R_{j,j}$, it will not perturb the following matrix at all. Let

$$T_n(c) = \operatorname{diag}(1, s, \ldots, s^{n-1}) \begin{bmatrix} 1 & -c & -c & \cdots & -c \\ 0 & 1 & -c & \cdots & -c \\ & \ddots & & \vdots & \vdots \\ \vdots & & & 1 & -c \\ 0 & & \ldots & 0 & 1 \end{bmatrix}$$

with $c^2 + s^2 = 1$ with $c, s > 0$. For $n = 100$ the smallest diagonal value of $T_n(0.2)$ is 0.13, but its smallest singular value is $O(10^{-8})$ [22].

Better condition estimators will not fail on this example, but they may fail on others. It is relatively easy to safeguard our algorithm against failures of the estimator. A few inverse iterations on $R^*R$ will reliably estimate the smallest singular

value. Inverse iteration is cheap because $R$ is triangular. If we find that $R$ is still ill conditioned, we add more rows to $B$ and rotate them into $R$ using Givens rotations. The resulting factorization remains backward stable.

To find a perturbation that will reduce the number of tiny singular values, we find an approximation of the smallest singular value and a corresponding right singular vector of $R$. Suppose that $\sigma$ and $v$ are such a pair, with $||v||_2 = 1$ and $||Rv||_2 = \sigma$. Let $i$ be the index of the largest absolute value in $v$. Since $||v||_2 = 1$ we must have $|v_i| \geq 1/\sqrt{n}$. We add to $B$ a row $b^*$:

$$b_j = \begin{cases} c_A, & j = i, \\ 0, & j \neq i. \end{cases}$$

We now have

$$\begin{aligned} \left\| \begin{bmatrix} R \\ b^* \end{bmatrix} v \right\|_2 &\geq \|b^* v\|_2 \\ &= |b^* v| \\ &\geq c_A v_i \\ &\geq c_A/\sqrt{n}. \end{aligned}$$

If $\tau \geq n + 1$, then

$$\left\| \begin{bmatrix} R \\ b^* \end{bmatrix} v \right\|_2 \geq \frac{\sqrt{n+1}c_A}{\tau},$$

and the number of singular values that are smaller than $\sqrt{n+1}c_A/\tau$ is reduced by one. We repeat the process until all singular values are large enough. If we estimate $c_A = ||A||_2$ directly (using power iteration), then the constraint on $\tau$ can be relaxed to $\tau > \sqrt{2n}$.

The combination of a less-than-perfect condition estimation with the kinds of perturbations that we use during the factorization (rows with a single nonzero) can potentially lead to a cascade of unnecessary perturbations. Suppose that the $j$th column of the matrix is dependent (or almost dependent) on the first $j - 1$ columns, but that the condition estimator missed this and estimated that the leading $j$-by-$j$ block of $R$ is well conditioned. Suppose further that after the factorization of column $j + 1$, the condition estimator finds that the leading $(j + 1)$-by-$(j + 1)$ block of $R$ is ill conditioned (it is). Our algorithm will perturb column $j + 1$, which does not improve the conditioning of $R$. This can keep on going. From now on, $R$ remains ill conditioned, so whenever the condition estimator finds that it is, our algorithm will perturb another column. These perturbations do not improve the conditioning, but they slow down the iterative solver. Situations like these are unlikely, but, in principle, they are possible. Therefore, we invoke the condition estimator before and after each perturbation. If a perturbation does not significantly improve the conditioning, we refrain from further perturbations. We will fix the ill conditioning by perturbing $R$ after it is computed (it may also be possible to use inverse iteration to produce a more reliable perturbation during the factorization rather than wait until it is complete).

**7. Numerical examples.** In this section we give simple numerical examples for the applications described in section 5. The goal is to illustrate the benefits of the tools developed in this paper.

**7.1. Dropping dense rows for sparsity; updating.** Consider the matrix

$$A = \begin{bmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \\ \beta_1 & \cdots & \beta_n \end{bmatrix}$$

for some (real or complex) $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$. Suppose that we want to find the least-squares solution to $\min_x ||Ax - b||_2$. The $R$ factor of the $QR$ factorization of $A$ will be completely full, because $A^*A$ is full. Therefore, solving the equation using the $QR$ factorization will take $\Theta(n^3)$ time. If the equation is solved using LSQR, then every iteration will cost $\Theta(n)$ operations, but the number of iterations done is proportional to $\kappa(A)$. The value of $\kappa(A)$ can be very large for certain values of $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$.

Our analysis suggests a new method for solving the problem. We can remove the last row of $A$ and form the preconditioner $R = \mathrm{diag}(\alpha_1, \ldots, \alpha_n)$. Our analysis shows that when solving the equation using LSQR preconditioned by $R$, only 2 iterations will be done. Each iteration still costs $\Theta(n)$ operations, amounting to a linear time algorithm for solving the equation. In general, if there are $m \ll n$ full rows, an application of LSQR with a preconditioner that is only the diagonal will converge in $m$ iterations, each of them with $\Theta(nm)$ operations. The total running time will be $\Theta(nm^2)$, while regular LSQR will complete in $\Theta(n^2m)$ operations, and a $QR$-based algorithm will complete in $\Theta(n^3)$ operations. With Heath's method [23] the total running time will be $\Theta((n+m)m^2)$. We conducted experiments that validate this analysis.

**7.2. Adding rows to solve rank deficient problems.** Consider the matrix $A$ and vector $b$ generated by the following commands in MATLAB:

```
rand('state', 0);
m = ceil(n/4);
A0 = rand(n, m);
[U,Sigma1,V] = svd(A0, 0);
Sigma = diag(10 .^ [linspace(1, -4, m-1) -12]);
A1 = U*Sigma*V';
A = [A1 rand(n, m)];
b = rand(m, 1);
```

The code builds an $n \times \frac{n}{2}$ matrix $A$, which is ill conditioned ($\kappa \approx 10^{12}$ and norm around 1). We wish to solve the least squares problem $\min \|Ax - b\|_2$. The matrix is built so that column $n/4$ is close to a linear combination of the columns to its left. The first command resets the random number generator so that each run will generate the same matrix and vector.

A $QR$ factorization without pivoting generates a very small diagonal value (around $10^{-12}$) in position $(\frac{n}{4}, \frac{n}{4})$ of $R$. Using the factorization to solve $\min_x \|Ax - b\|_2$ leads to a solution with norm around $10^{11}$. In many cases, the desired solution is the minimizer in the subspace that is orthogonal to right singular vectors of $A$ that correspond to singular values around $10^{-12}$ and smaller. We refer to such a solution as a *truncated solution*. A $QR$ factorization without pivoting is useless for finding the truncated solution or an approximation of it.

One way to compute a low-norm almost-minimizer of the truncated problem is to use a rank revealing $QR$. If $A$ is dense (as in our example), this is an effective

solution. Rank revealing $QR$ factorization algorithms have also been developed for sparse matrices, but they are complex and sometimes expensive (since they cannot control sparsity as well as non–rank revealing algorithms) [12, 30, 5].

In our example, running LSQR with a convergence threshold of $r = 10^{-10}$ (for $n = 100$) led to an acceptable solution (with norm around $10^3$). With $r = 10^{-15}$, LSQR returned a solution with norm $10^{11}$, which is clearly not a good truncated solution. Due to the ill-conditioning of $A$, many iterations are required for LSQR to converge. Even with $r = 10^{-10}$, LSQR converged slowly, taking 423 iterations to converge.

We propose using instead the algorithm described in section 6 to generate an effective preconditioner that allows LSQR to solve the truncated problem. We generated two preconditioners using the two versions of our algorithm, one with $c_A = ||A||_1$ and the other with $c_A = ||A||_2$. We set the threshold $\tau$ to $10^{10}$. In both cases a single row was added with a single nonzero in column 25. The need to add a row was detected, in both cases, using the incremental condition estimator during the initial $QR$ factorization. With $c_A = ||A||_1$ the condition number of the factor was $\kappa(R) \approx 3.41 \times 10^5$, while with $c_A = ||A||_2$ the condition number was $\kappa(R) \approx 1.78 \times 10^5$. When using $R$ as a preconditioner to LSQR with threshold $10^{-10}$, a single iteration was enough to converge in both cases. The norm of the minimizer $x$ was of order $10^3$ in both cases.

The different methods that produced solutions with norm around $10^3$ produced different solution vectors $x$ with slightly different norms (even the two preconditioned LSQR methods). To see why, let $v$ be the singular vector that corresponds to the singular value $10^{-12}$. LSQR uses the norm $||A^*(Ax - b)||_2$ as a stopping criterion. Adding $\rho v$ to a vector $x$ changes $||A^*(Ax - b)||_2$ by at most $\rho \times 10^{-24}$, so even a large $\rho$ rarely affects this stopping criterion. Therefore, different methods can return different solutions, say, $x$ and $x + \rho v$, possibly with $\rho \gg \|x\|$. Such solutions are very different from each other but with norms and residual norms that both differ by at most $\rho \times 10^{-12}$. Both solutions are good, but they are different; this is a reflection of the ill-conditioning of the problem.

**8. Conclusions.** This paper presented a theoretical analysis of certain preconditioned least-squares solvers. The solvers use a preconditioner that is related to a low-rank perturbation of the coefficient matrix. The perturbation can be the result of an updating or downdating (following the computation of a preconditioner or a factor of the original coefficient matrix), of dropping dense rows, or of an attempt to make the preconditioner well conditioned when the coefficient matrix is ill conditioned or rank deficient. We note that further research is required to determine how to drop rows effectively in sparse $QR$ factorizations; here we gave only evidence that this idea can be effective, but we did not provide a row-dropping algorithm.

The paper also proposed a specific method to perturb a $QR$ factorization of an ill-conditioned or rank deficient matrix.

Our theoretical analysis uses a novel approach: we count the number of generalized eigenvalues that move away from a cluster of eigenvalues (sometimes consisting only of the value 1) due to perturbations. This allows us to bound the number of iterations in iterative least-squares solvers like LSQR, which are implicit versions of conjugate gradients on the normal equations.

This approach complements the more common way of bounding iteration counts in optimal Krylov-subspace solvers, which is based on bounding the condition number of the preconditioned system.

We have also presented limited experimental results, which are meant to illustrate

the use of the techniques rather than establish their effectiveness or efficiency. We plan to design and implement a sparse $QR$ factorization code that will incorporate these techniques, but this is beyond the scope of this paper. Once we have an implementation for the sparse case, we plan to perform extensive testing of the technique that this paper analyzes theoretically.

**Acknowledgments.** This research was partially motivated by discussions with Jason Riedy concerning theoretical analysis of diagonal perturbations in sparse $LU$ factorizations. We thank Jason for these discussions. Thanks to Michael Saunders for pointing out [19] to us. We also would like to thank the referees for their valuable comments.

## REFERENCES

[1] H. Avron, E. Ng, and S. Toledo, *A Generalized Courant-Fischer Minimax Theorem*, Tech. report, Tel-Aviv University, Tel-Aviv, Israel, 2008.
[2] J. L. Barlow and U. B. Vemulapati, *Rank detection methods for sparse matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1279–1297.
[3] C. Bischof and P. Tang, *LAPACK Working Note* 33: *Robust Incremental Condition Estimation*, Tech. report, University of Tennessee, Knoxville, TN, 1991.
[4] C. H. Bischof, *Incremental condition estimation*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 312–322.
[5] C. H. Bischof and P. C. Hansen, *Structure-preserving and rank-revealing QR-factorizations*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1332–1350.
[6] C. H. Bischof, J. G. Lewis, and D. J. Pierce, *Incremental condition estimation for sparse matrices*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 644–659.
[7] Å. Björck, *A general updating algorithm for constrained linear least squares problems*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 394–402.
[8] Å Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
[9] I. Brainman and S. Toledo, *Nested-dissection orderings for sparse LU with partial pivoting*, in Proceedings of the 10th SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, 2001, CD-ROM, SIAM, Philadelphia, 2001.
[10] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra Appl., 88/89 (1987), pp. 67–82.
[11] T. F. Chan and P. C. Hansen, *Some applications of the rank revealing QR factorization*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 727–741.
[12] S. Chandrasekaran and I. C. F. Ipsen, *On rank-revealing factorisations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 592–622.
[13] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.
[14] T. A. Davis, *Direct Methods for Sparse Linear Systems*, Fundamentals of Algorithms 2, SIAM, Philadelphia, 2006.
[15] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng, *A Column Approximate Minimum Degree Ordering Algorithm*, Tech. report TR-00-005, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 2000.
[16] I. S. Duff and C. Vömel, *Incremental norm estimation for dense and sparse matrices*, BIT, 42 (2002), pp. 300–322.
[17] L. V. Foster, *The probability of large diagonal elements in the QR factorization*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 531–544.
[18] A. George and M. T. Heath, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69–83.
[19] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright, *On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method*, Math. Programming, 36 (1986), pp. 183–209.
[20] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.
[21] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[23] M. T. Heath, *Some extensions of an algorithm for sparse linear least squares problems*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 223–237.
[24] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.
[25] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
[26] S.-M. Lu and J. L. Barlow, *Multifrontal computation with the orthogonal factors of sparse matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 658–679.
[27] E. Ng, *A scheme for handling rank-deficiency in the solution of sparse linear least squares problems*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1173–1183.
[28] M. K. Ng, *Iterative Methods for Toeplitz Systems (Numerical Mathematics and Scientific Computation)*, Oxford University Press, New York, 2004.
[29] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[30] D. J. Pierce and J. G. Lewis, *Sparse multifrontal rank revealing QR factorization*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 159–180.
[31] P.Å. Wedin, *Perturbation theory for pseudo-inverses*, BIT, 13 (1973), pp. 217–232.