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 $\ddot{A}$ of A. More precisely, we show that the R factor of $\ddot{A}$ 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 $\ddot{A}$. 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 $\ddot{A}$. 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.

1. Introduction

This paper shows that the R factor from the QR factorization of a perturbation $\ddot{A}$ of a matrix A is an effective least-squares preconditioner for A. More specifically, we show 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 $\ddot{A}$ is still an effective preconditioner if $\ddot{A}$ 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 $\ddot{A}^*\ddot{A}$ (where $\ddot{A}^*$ is the adjoint of $\ddot{A}$), in the sense that the generalized condition number of $(\ddot{A}^*\ddot{A}, M)$ 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 $\ddot{A}$; we can use M as a preconditioner instead.

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

Date: July 2007.

of *runaway* eigenvalues, which are the ones that are not 1 (or outside the interval), which guarantees rapid convergence of LSQR.

These results, along with our algorithm for perturbing a matrix to improve its conditioning, have several applications. They allow us to drop rows from a sparse A to increase the sparsity of R. They allow us to solve updated and downdated least-squares problems efficiently without recomputing the factor or the preconditioner. They allow us to solve what-if scenarios (least-squares problems modified by fixing the value of a few unknowns). They allow us to solve rank-deficient least-squares problem without a rank-revealing factorization.

In this paper we do not systematically address the cost of constructing preconditioners or factors of perturbed matrices. We focus on the spectral theory and leave detailed performance comparisons to future research.

Some of these solution techniques are not new. It appears that some of them have been used by practitioners (see, e.g., [19]) without a theoretical analysis.

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).

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 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 [7]. 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, 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 smaller than with strict pivoting [12, 30]. Some algorithms [5] 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, 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. This 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 λ is a *finite generalized eigenvalue* of the matrix pencil (pair) (S, T) if there is a vector $v \neq 0$ such that

$$Sv = \lambda T v$$

and $Tv \neq 0$. We say that ∞ is a *infinite generalized eigenvalue* of (S, T) if there exist a vector $v \neq 0$ such that $Tv = 0$ but $Sv \neq 0$. Note that ∞ 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 λ.

Throughout the paper eigenvalues are ordered from smallest to largest. We will denote the kth eigenvalue of S by $\lambda_k(S)$, and the kth 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 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 the Appendix. 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 an 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 an 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. For completeness the proof appears in the Appendix.

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

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

and

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

4. Spectral Theory

The generalized spectrum of (A^*A, A^*A) is very simple: the pencil has $\operatorname{rank}(A)$ 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.

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 $1 \le k + r < n$. Define*

$$\chi = \begin{bmatrix} B \\ C \end{bmatrix}.$$

The following claims hold:

(1) *In the pencil $(A^*A, A^*A + B^*B - C^*C)$, at most* $\operatorname{rank}(\chi) \le k + r$ *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 (a) the pencil $(A^*A, A^*A+B^*B-C^*C)$ does not have indeterminate eigenvectors, (b) the multiplicity*

of the eigenvalue 1 *is exactly* $\dim \operatorname{null}(\chi) \geq n-k-r$*, and (c) the multiplicity of the zero eigenvalue is exactly* $\dim \operatorname{null}(A)$.

(3) *The sum pencil* (A^*A, A^*A+B^*B) *cannot have an infinite eigenvalue and all its eigenvalues are in the interval* $[0,1]$.

Proof. First, notice that $v \in \operatorname{null}(\chi)$ if and only if $v \in \operatorname{null}(B) \cap \operatorname{null}(C)$. 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 λ, then the relationship of v to the null spaces of A and the relationship of B^*Bv to C^*Cv, determine λ in the following way:

	$v \in \operatorname{null}(A)$	$v \notin \operatorname{null}(A)$
$B^*Bv = C^*Cv$	indeterminate	$\lambda = 1$
$B^*Bv \neq C^*Cv$	$\lambda = 0$	$\lambda \neq 0$ and $\lambda \neq 1$

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

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

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

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

If $v \in \operatorname{null}(A)$ 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 $v \notin \operatorname{null}(A)$ and $B^*Bv \neq C^*Cv$, then λ 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 $v \in \operatorname{null}(B) \cap \operatorname{null}(C) = \operatorname{null}(\chi)$ then clearly $B^*Bv = C^*Cv$. So, if v is a determined generalized eigenvalue corresponding to a eigenvalue different from 1, then $v \notin \operatorname{null}(\chi)$. Therefore, the dimension of the space spanned by these vectors is bounded by $\operatorname{rank}(\chi) \leq k+r$, 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). Since $A^*A+B^*B-C^*C$ is full rank then for every vector $v \neq 0$ we have $(A^*A+B^*B-C^*C)v \neq 0$, so vector v cannot be an indetermined eigenvector of $(A^*A, A^*A+B^*B-C^*C)$, and the pencil has no indetermined eigenvalues. The multiplicity of the eigenvalue 1 follows from the fact that if v is a generalized eigenvector corresponding to 1 then we must have $B^*Bv = C^*Cv$. Since 1 is not a generalized eigenvalue of the pencil (B^*B, C^*C) then we must have $B^*Bv = C^*Cv = 0$. Therefore, the space of eigenvectors corresponding to 1 is exactly $\operatorname{null}(\chi)$. The multiplicity of the eigenvalue 0 follows from the fact that every $0 \neq v \in \operatorname{null}(A)$ 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 eigenvecctor. The converse is true from the table.

We now show that Claim 3 holds. In the sum pencil (A^*A, A^*A+B^*B), λ cannot be infinite. Suppose for contradiction that it is. Then $(A^*A+B^*B)v=0$ but $A^*Av\neq 0$. We get $v^*(A^*A+B^*B)v=0$, but $v^*(A^*A+B^*B)v=v^*A^*Av+v^*B^*Bv>0$. To show that the generalized eigenvalues are in the range $[0,1]$, notice that if $\lambda\neq 0$ is a finite generalized eigenvalue of (A^*A, A^*A+B^*B) then there exist a vector $v\neq 0$ such that

$$\begin{aligned}\lambda &= \frac{v^*A^*Av}{v^*(A^*A+B^*B)v}\\ &= \frac{v^*A^*Av}{v^*A^*Av+v^*B^*Bv}\,.\end{aligned}$$

Since both v^*A^*Av and v^*B^*Bv are greater than or equal to 0 the claim follows immediately.

For completeness, we also characterize the infinite eigenvectors of the pencil $(A^*A, A^*A+B^*B-C^*C)$. Such an eigenvector v satisfies

$$\begin{aligned}A^*Av &\neq 0\\ (A^*A+B^*B-C^*C)\,v &= 0\,,\end{aligned}$$

or

$$(A^*A+B^*B)v=C^*Cv\neq 0\,.$$

Thus, v is a generalized eigenvector of (A^*A+B^*B, C^*C) corresponding to the eigenvalue 1. $\square$

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

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

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

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 non-runaways are not necessarily at 1: they can move about the interval whose size determines the condition number of the original preconditioned system.

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 an Hermitian positive semidefinite matrix. Suppose that* $\operatorname{null}(M) \subseteq \operatorname{null}(A^*A)$, $\operatorname{null}(M) \subseteq \operatorname{null}(B^*B)$ *and* $\operatorname{null}(M) \subseteq \operatorname{null}(C^*C)$. *If*

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

then

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

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

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

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

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

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

□

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 an Hermitian positive semidefinite matrix, such that* $\operatorname{null}(M) \subseteq \operatorname{null}(A^*A)$ *and* $\operatorname{null}(M) \subseteq \operatorname{null}(\tilde{A}^*\tilde{A})$*. Suppose that*

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

Then we have

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

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

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

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

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

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

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

Since, by the second equality of the Theorem 3.4,

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

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 $x \in U$,

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

Since, by the first equality of the 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 $\lambda_{t-k}(\tilde{A}^*\tilde{A}, M) \le \lambda_{t-r}(\tilde{A}^*\tilde{A}, M) \le \beta$. $\square$

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 an Hermitian matrix and that $T \in \mathbb{C}^{n\times n}$ is an Hermitian positive definite matrix. For all $1 \le k \le n$ we have*

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

Proof. We will use Theorem 3.4, by presenting, for each k, a subspace U_k and V_k that bound $\lambda_k(S,T)$. Let $u_1, \ldots, u_n$ be a set of orthonormal eigenvectors corresponding to $\lambda_1(S), \ldots, \lambda_n(S)$. Define $U_k = \mathrm{span}\{u_1, \ldots, u_k\}$. For every $x \in U_k$ we can write

$$x = \alpha_1 u_1 + \cdots + \alpha_k u_k,$$

therefore

$$x^*Sx = \sum_{i=1}^{k} \alpha_i^2 \lambda_i(S) \le \lambda_k(S) \sum_{i=1}^{k} \alpha_i^2$$

and

$$x^*x = \sum_{i=1}^{k} \alpha_i^2.$$

Combining the last two we get

$$\frac{x^*Sx}{x^*x} \le \lambda_k(S).$$

Also,

$$\frac{x^*x}{x^*Tx} = \frac{1}{x^*Tx/x^*x} \le \frac{1}{\min_{x\in\mathbb{R}^n} x^*Tx/x^*x} = \frac{1}{\lambda_1(T)}.$$

Combining the two we get

$$\frac{x^*Sx}{x^*Tx} = \frac{x^*Sx}{x^*x} \cdot \frac{x^*x}{x^*Tx} \le \frac{\lambda_k(S)}{\lambda_1(T)}.$$

By Theorem 3.4,

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

Since $\dim(U_k) = k$ and $\max_{x\in U_k} x^*Sx/x^*Tx \le \lambda_k(S)/\lambda_1(T)$ we have shown the upper bound on $\lambda_k(S,T)$.

For the lower bound, define $V_k = \mathrm{sp}\{u_k, \ldots, u_n\}$ and use the second inequality of the Theorem 3.4 in steps similar to the ones above. $\square$

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 $1 \le k < n$. 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 $\lambda_n(T) \ge \lambda_n(S) \ge \alpha$. By the lemma,

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

For any k such that $\lambda_k(S) \le \epsilon\alpha$, 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 $1 \le k < n$. Assume that A^*A+B^*B is full rank. Denote $\alpha = ||A^*A||_2$ and suppose that $||B^*B||_2 \le \gamma\alpha$. 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 $\lambda_n(T) \le ||A^*A||_2 + ||B^*B||_2 \le (1+\gamma)\alpha$. Therefore

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

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

The theorems show that the numerical rank of the preconditioned system is the same as of the original system, up to an appropriate relaxation of the rank threshold. Suppose that after the dth eigenvalue there is a big gap. That is, there are d eigenvalues of A^*A are smaller than $\epsilon\alpha$, and the remaining $n-d$ are larger than $\eta\alpha$, where α is the largest eigenvalue of A^*A. The ratio between the largest eigenvalue and the dth 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 $||B^*B||_2 \le m||A^*A||_2$ and $\kappa(A^*A + B^*B) \le \tau^2$, where $\tau \ge n+1$ is a given threshold, and a slightly more expensive algorithm that only requires $\tau \ge \sqrt{2n}$ and guarantees $||B^*B||_2 \le ||A^*A||_2$.

5. Applications To Least-Square 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, 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.

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 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).

We do not currently have a sophisticated algorithm to decide when to drop rows. That is, even if the remaining rows are very sparse, the R factor can be fairly dense. If this is the case, then dropping the densest rows increases the number of iterations without decreasing the sparsity of R. Without a sophisticated dropping algorithm, we recommend that rows with density exceeding a certain threshold (say 25%) be dropped.

It is interesting to compare our method to the method used by Heath in [23] to handle dense rows (see also [8] and [28]). Suppose that the last k rows in $A \in \mathbb{C}^{(m+k)\times n}$ are dense. Our algorithm factors the first m rows and performs at most $k + 1$ iterations, each involving the application of A, A^*, R^{-1} and $(R^{-1})^*$. Each iteration performs $\Theta(|A| + |R|)$ operations, where $|A|$ is the number of non-zeros in A and $|R|$ is the number of non-zeros in R. The total amount of work in addition to the factorization is $\Theta(k(|A| + |R|))$. Since A has k dense rows, each application of A and A^* costs at least $2nk$ arithmetic operations.

The dominant costs in Heath's method are the factorization of the first m rows of A (same as in our approach), k triangular solves with R^*, and a dense QR factorization of an $(n+k)$-by-k matrix. The total amount of work, apart from the initial sparse factorization, is $\Theta(k|R| + (n+k)k^2)$ operations. If R is denser than the first m rows of A, or if $|A| = O(nk)$, the asymptotic cost of the two methods is similar; this appears to be the common case. Otherwise, our method is more expensive due to the repeated application of A and A^*.

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 involved 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, 1, 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 can be used as a preconditioner when applying LSQR to the system. The maximum number of iterations that LSQR will perform when using exact arithmetic is the number of rows in B. Our experiments have shown that in practice the number is usually even smaller, but sometimes is slightly larger. It is important to set the threshold parameter of LSQR to the threshold from which singular values are to be truncated. Our numerical experiments have shown that when the threshold parameter is set this way, the solution formed by LSQR when stopped by this criteria is a small-norm solution to the desired truncated problem (but not necessarily the minimum-norm solution). An explanation why this is so can be found in [29]. This is why the results in subsection 4.3 are important: the fact that the spectral gap between truncated and non-truncated singular values is maintained means that the algorithm finds a good truncated solution.

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 $\min_x \|\bar{A}x - b\|_2$, where $\bar{A}$ has the same singular value decomposition as A except that the k smallest eigenvalues are truncated to 0.

We start with a simple lemma that forms the basis to 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 $\dim \operatorname{null}(A) = n - \operatorname{rank}(A) = k$, and the multiplicity of the 1 eigenvalue of $(A^{*}A, R^{*}R)$ is exactly $n - \operatorname{rank}(B) = n - k$. 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 minimizers 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* $\operatorname{rank}(A) = n - k$ *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 $P = \left[\begin{array}{cc} I_{m\times m} & 0_{m\times k} \end{array}\right]$. 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 Corollary 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 $\left[\begin{smallmatrix} \bar{A} \\ B \end{smallmatrix}\right] = \bar{Q}\bar{R}$, 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 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 singular value decomposition as A except that the k smallest eigenvalues are truncated to 0. Denote*

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

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

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

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

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

and

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

Then, provided that $\delta < 1$,

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

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 δ is small (which happens when C is well conditioned and A has k tiny singular values) and R is not ill conditioned and not too large, and 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 δ, 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 $\|\Delta C\|_2 \le \sigma_{n-k}(A)$. If we define $\epsilon = \sigma_{n-k}(A)/\|C\|_2$ then $\kappa(C)\epsilon = \delta < 1$ and $\|\Delta C\|_2 \le \epsilon\|C\|_2$. We can apply a variant of result from Wedin [31] (see Theorem 20.1 in [25] for the specific version that we use) and conclude that

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

□

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 only adds 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 τ, 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) $\kappa(R) \le \tau$, 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 non-zeros 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 non-zero, which we set to $\pm c_A$. Each row of B has a different nonzero column. It follows that B^*B is diagonal with

$||B||_2 = c_A$. 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 only need 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

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

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 backwards 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 eigenvector of R^*R. Suppose that σ and v are such a singular 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^*,

$$b_j = \begin{cases} c_A \cdot \mathrm{sign}\left((Rv)_i\right) & j = i \\ 0 & j \neq i \end{cases}$$

We now have

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

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 τ 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 jth 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

$$R = \begin{bmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{bmatrix}.$$

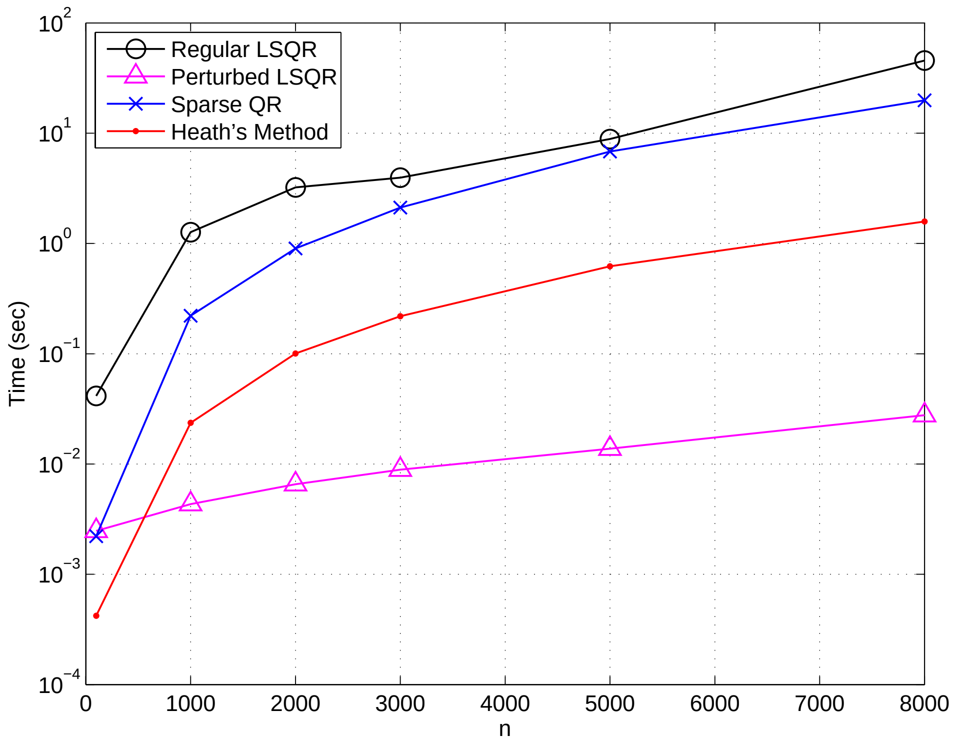

FIGURE 1. Running time when solving the least spares equation $\min_x ||Ax - b||_2$, when consists of a diagonal and a dense row. We measured all methods for an increasing matrix size. The x-axis is the number of columns in the matrix. The diagonal and dense row values are chosen randomly.

Our analysis shows that when solving the equation using LSQR preconditioned by R only 2 iterations will be done. Each iteration still cost $\Theta(n)$ 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 a $\Theta(n^3)$ operations. With Heath's method [23] the total running time will be $\Theta((n+m)m^2)$.

The same analysis also applies to updating problems. If we are given A without its last row and compute the R factor of the input matrix, we can use this factor for efficiently solving least-squares problems involving an additional arbitrary constraint.

We conducted a simple experiment to test the actual convergence time using MATLAB 7.2 [27]. We generated $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ using `rand` for various values of n. We then ran all the algorithms, setting the tolerance of LSQR to 10^{-6}, and measured the solution time using MATLAB's internal timer. The results are shown in Figure 1. It is easy to see that our method is superior to the other methods in this case.

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 codes builds a $n \times \frac{n}{2}$ 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 in a special way 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 in 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 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].

LSQR compute a low-norm almost-minimizer of the truncated problem. When using LSQR, it is important to carefully set its convergence threshold. Let

$$r = \frac{\sigma_{\text{truncation}}}{\sigma_{\max}}$$

be the ratio of the largest singular value that we want to truncate to the maximal singular value of A. When the LSQR convergence threshold is larger than r, LSQR computes an approximate solution to a problem in which all the singular values smaller than $\sigma_{\text{truncation}}$ have been truncated [29]. In our example, setting $n = 100$ and $r = 10^{-10}$ led to an acceptable solution (with norm around 10^3). With $r = 10^{-15}$, LSQR returned a solution with norm 10^{11}, clearly not a good truncated solution. Due to the ill-conditioning of A many iterations will be required for LSQR to converge. Even with $r = 10^{-10}$, LSQR converged slowly, taking 423 iterations to converge.

We propose to use 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

τ 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 ρv to a vector x changes $||A^*(Ax-b)||_2$ by at most $\rho \times 10^{-24}$, so even a large ρ 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 theoretical analysis of certain preconditioned least-square 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.

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 to 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.

Acknowledgement. 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 [19] to us.

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), and by grant 2002261 from the United-States-Israel Binational Science Foundation.

References

[1] Jesse L. Barlow and Udaya B. Vemulapati. Rank detection methods for sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 13(4):1279–1297, 1992.

[2] Denis S. Bernstein. *Matrix Mathematics: Theory, Facts, and Formulas with Applications to Linear Systems Theory.* Princeton University Press, 2005.

[3] C. Bischof and P. Tang. Robust incremental condition estimation. Technical report, Knoxville, 1991.

[4] Christian H. Bischof. Incremental condition estimation. *SIAM Journal on Matrix Analysis and Applications*, 11(2):312–322, 1990.

[5] Christian H. Bischof and Per Christian Hansen. Structure-preserving and rank-revealing QR-factorizations. *SIAM Journal on Scientific and Statistical Computing*, 12(6):1332–1350, 1991.

[6] Christian H. Bischof, John G. Lewis, and Daniel J. Pierce. Incremental condition estimation for sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 11(4):644–659, 1990.

[7] Å Björck. *Numerical Methods for Least Squares Problems.* SIAM, 1996.

[8] Ake Bjorck. A general updating algorithm for constrained linear least squares problems. 5:394–402, 1984.

[9] Igor Brainman and Sivan Toledo. Nested-dissection orderings for sparse LU with partial pivoting. In *Proceedings of the 10th SIAM Conference on Parallel Processing for Scientific Computing*, Norfolk, Virginia, March 2001. 10 pages on CDROM.

[10] T. F. Chan. Rank revealing QR factorizations. 88/89:67–82, 1987.

[11] Tony F. Chan and Per Christian Hansen. Some applications of the rank revealing QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 13(3):727–741, 1992.

[12] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. On rank-revealing factorisations. *SIAM Journal on Matrix Analysis and Applications*, 15(2):592–622, 1994.

[13] Paul Concus, Gene H. Golub, and Dianne P. O'Leary. A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In James R. Bunch and Donald J. Rose, editors, *Sparse Matrix Computations*, pages 309–332. Academic Press, New York, 1976.

[14] Thimothy A. Davis. *Direct Methods for Sparse Linear Systems.* SIAM, Philadelphia, 2006.

[15] Timothy A. Davis, John R. Gilbert, Stefan I. Larimore, and Esmond G. Ng. A column approximate minimum degree ordering algorithm. Technical Report TR-00-005, Department of Computer and Information Science and Engineering, University of Florida, 2000.

[16] Ian S. Duff and Christof Vömel. Incremental norm estimation for dense and sparse matrices. *BIT Numerical Mathematics*, (2):300–322, 2002.

[17] Leslie V. Foster. The probability of large diagonal elements in the QR factorization. *SIAM Journal on Scientific and Statistical Computing*, 11(3):531–544, 1990.

[18] A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. 34:69–83, 1980.

[19] Philip E. Gill, Walter Murray, Michael A. Saunders, J. A. Tomlin, and Margaret H. Wright. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Math. Program.*, 36(2):183–209, 1986.

[20] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 7:206–216, 1965.

[21] G. H. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. 2:205–224, 1965.
[22] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
[23] Michael T. Heath. Some extensions of an algorithm for sparse linear least squares problems. *SIAM Journal on Scientific and Statistical Computing*, 3(2):223–237, 1982.
[24] M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *National Bureau of Standards Jounal of Research*, 49:409–436, 1952.
[25] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 1996.
[26] Szu-Min Lu and Jesse L. Barlow. Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 17(3):658–679, 1996.
[27] The MathWorks. Matlab version 7.2. software package, January 2006.
[28] Esmond Ng. A scheme for handling rank deficiency in the solution of sparse linear least squares problems. 12:1173–1183, 1991.
[29] Christopher C. Paige and Michael A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8(1):43–71, 1982.
[30] Daniel J. Pierce and John G. Lewis. Sparse multifrontal rank revealing QR factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):159–180, 1997.
[31] P.Å. Wedin. Perturbation theory for pseudo-inverses. 13:217–232, 1973.

Appendix: Proof of the Generalized Courant-Fischer Minimax Theorem

We begin by stating and proving a generalization of the Courant-Fischer Theorem for pencils of Hermitian positive definite matrices.

Theorem 8.1. *Let $S, T \in \mathbb{C}^{n\times n}$ be Hermitian matrices. If T is also positive definite then*

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

and

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

Proof. Let $T = L^*L$ be the Cholesky factorization of B. Let U be some k-dimensional subspace of $\mathbb{C}^n$, let $x \in U$, and let $y = Lx$. Since T is Hermitian positive definite (hence nonsingular), the subspace $W = \{Lx : x \in U\}$ has dimension k. Similarly, for any k-dimensional subspace W, the subspace $U = \left\{L^{-1}x : x \in W\right\}$ has dimension k. We have

$$\frac{x^*Sx}{x^*Tx} = \frac{x^*L^*L^{-*}SL^{-1}Lx}{x^*L^*Lx} = \frac{y^*L^{-*}SL^{-1}y}{y^*y}\,.$$

By applying the Courant-Fischer to $L^{-*}SL^{-1}$, we obtain

$$\begin{aligned}\lambda_k(L^{-*}SL^{-1}) &= \min_{\dim(W)=k} \max_{y\in W} \frac{y^*L^{-*}SL^{-1}y}{y^*y} \\ &= \min_{\dim(U)=k} \max_{x\in S} \frac{x^*Sx}{x^*Tx}\,.\end{aligned}$$

The generalized eigenvalues of (S,T) are exactly the eigenvalues of $L^{-*}SL^{-1}$ so the first equality of the theorem follows. The second equality can be proved using a similar argument. □

Before proving the generalized version of the Courant-Fischer Minimax Theorem we show how to convert an Hermitian positive semidefinite problem to an Hermitian positive definite problem.

Lemma 8.2. *Let $S, T \in \mathbb{C}^{n\times n}$ be Hermitian matrices. Assume that T is also a positive semidefinite and that $\mathrm{null}(T) \subseteq \mathrm{null}(S)$. For any $Z \in \mathbb{C}^{n\times \mathrm{rank}(T)}$ with range(Z) = range(T), the determined generalized eigenvalues of (S, T) are exactly the generalized eigenvalues of (Z^*SZ, Z^*TZ).*

Proof. We first show that Z^*TZ has full rank. Suppose that $Z^*TZv = 0$. We have $TZv \in \mathrm{null}(Z^*)$. Therefore, $TZv \perp \mathrm{range}(Z) = \mathrm{range}(T)$. Obviously $TZv \in \mathrm{range}(T)$, so we must have $v = 0$. Since $\mathrm{null}(Z^*TZ) = \{0\}$, the matrix Z^*TZ has full rank.

Suppose that λ is a determined eigenvalue of (S, T). We will show that it is a determined eigenvalue of (Z^*SZ, Z^*TZ). The pencil (Z^*SZ, Z^*TZ) has exactly $\mathrm{rank}(Z^*TZ)$ determined eigenvalues. We will show that Z^*TZ is full rank, so the pencil (Z^*SZ, Z^*TZ) has exactly $\mathrm{rank}(T)$ eigenvalues. Since the pencil (S, T) has exactly $\mathrm{rank}(T)$ determined eigenvalues, each of them an eigenvalue of (Z^*SZ, Z^*TZ), this will conclude the proof.

Now let μ be an eigenvalue of (Z^*SZ, Z^*TZ). It must be determined, since Z^*TZ has full rank. Let y be the corresponding eigenvector, $Z^*SZy = \mu Z^*TZy$, and let $x = Zy$. Now there are two cases. If $\mu = 0$, then $SZy = Sx = 0$ (since Z^* has full rank and at least as many columns as rows). The vector x is in $\mathrm{range}(Z) = \mathrm{range}(T)$, $Tx \neq 0$. This implies that $\mu = 0$ is also a determined eigenvalue of (S, T).

If $\mu \neq 0$, the analysis is a bit more difficult. Clearly, $TZy \in \mathrm{range}(T) = \mathrm{range}(Z)$. But $\mathrm{range}(Z) = \mathrm{range}(Z^{*+})$ [2, Proposition 6.1.6.vii], so $Z^{*+}Z^*TZy = TZy$ [2, Proposition 6.1.7]. We claim that $SZy \in \mathrm{range}(Z)$. If it is not, it Zy must be in $\mathrm{null}(T) \subseteq \mathrm{null}(S)$, but μ would have to be zero. Therefore, we also have $Z^{*+}Z^*SZy = SZy$, so by multiplying $Z^*SZy = \mu Z^*TZy$ by Z^{*+} we see that μ is an eigenvalue of (S, T). □

We are now ready to prove Theorem 3.4, the generalization of the Courant-Fischer Minimax Theorem. The technique is simple: we use Lemma 8.2 to reduce the problem to a smaller-sized full-rank problem, apply Theorem 8.1 to characterize the determined eigenvalues in terms of subspaces, and finally show a complete correspondence between the subspaces used in the reduced pencil and subspaces used in the original pencil.

Theorem. *(Generalized Courant-Fischer Minimax Theorem) Let $S, T \in \mathbb{C}^{n\times n}$ be Hermitian matrices. Assume that T is also a positive semidefinite and that $\mathrm{null}(T) \subseteq \mathrm{null}(S)$. For $1 \le k \le \mathrm{rank}(T)$ we have*

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

and

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

Proof. Let $Z \in \mathbb{C}^{n\times \mathrm{rank}(T)}$ have $\mathrm{range}(Z) = \mathrm{range}(T)$. We have

$$\lambda_k(S,T) = \lambda_k(Z^*SZ, Z^*TZ) = \min_{\dim(W)=k} \max_{x\in W} \frac{x^*Z^*SZx}{x^*Z^*TZx}$$

and

$$\lambda_k(S,T) = \lambda_k(Z^*SZ, Z^*TZ) = \max_{\dim(W)=\mathrm{rank}(T)-k+1} \min_{x\in W} \frac{x^*Z^*TZx}{x^*Z^*TZx} .$$

The leftmost equality in each of these equations follows from Lemma 8.2 and the rightmost one follows from Theorem 8.1.

We now show that for every k-dimensional subspace $U \subseteq \mathbb{C}^n$ with $U \perp \mathrm{null}(T)$, there exists a k-dimensional subspace $W \subseteq \mathbb{C}^{\mathrm{rank}(T)}$ such that

$$\left\{ \frac{x^*Sx}{x^*Tx} : x \in U \right\} = \left\{ \frac{y^*Z^*SZy}{y^*Z^*TZy} : y \in W \right\} ,$$

and vice versa. The validity of this claim establishes the min-max side of the theorem.

We first need to show that $k \le \mathrm{rank}(T)$. This is true because every vector in U is in $\mathrm{range}(T)$, so its dimension must be at most $\mathrm{rank}(T)$.

Define $W = \left\{ y \in \mathbb{C}^{\mathrm{rank}(T)} : Zy \in U \right\}$. Let $b_1, \ldots, b_k$ be a basis for U. Because $U \perp \mathrm{null}(T)$, $b_j \in \mathrm{range}(T)$, so there is a y_j such that $Zy_j = b_j$. Therefore, dimension of W is at most k. Now let the vectors y_i's be a basis of W and define $b_i = Zy_i$. The b_i's span U, so there are at most k of them, so the dimension of W is at least k. Therefore, it is exactly k.

Every $x \in U$ is orthogonal to $\mathrm{null}(T)$, so it must be in $\mathrm{range}(T)$. There exist a $y \in \mathbb{C}^{\mathrm{rank}(T)}$ such that $Zy = x$. So we have $x^*Sx/x^*Tx = y^*Z^*SZy/y^*Z^*TZy$. Combining with the fact that $y \in W$, we have shown inclusion of one side. Now suppose $y \in W$. Define $x = Zy \in U$. Again we have $x^*Sx/x^*Tx = y^*Z^*SZy/y^*Z^*TZy$, which shows the other inclusion.

Now we will show that for every k-dimensional subspace W there is a subspace U that satisfies the claim. Define $U = \{Zy : y \in W\}$. Because Z has full rank, $\dim(U) = k$. Also, $U \subseteq \mathrm{range}(Z) = \mathrm{range}(T)$ so $U \perp \mathrm{null}(T)$. The equality of the Raleigh-quotient sets follows from taking $y \in W$ and $x = Zy \in U$ or vice versa. $\square$

Current address: Haim Avron and Sivan Toledo: School of Computer Science, Tel-Aviv University, Esmond Ng: Lawrence Berkely National Laboratory