A Backward Stable Hyperbolic QR Factorization Method for Solving Indefinite Least Squares Problem

Hongguo Xu*

Dedicated to Professor Erxiong Jiang on the occasion of his 70th birthday.

Abstract

We present a numerical method for solving the indefinite least squares problem. We first normalize the coefficient matrix. Then we compute the hyperbolic QR factorization of the normalized matrix. Finally we compute the solution by solving several triangular systems. We give the first order error analysis to show that the method is backward stable. The method is more efficient than the backward stable method proposed by Chandrasekaran, Gu and Sayed.

Keywords. Indefinite least squares, hyperbolic rotation, $\Sigma_{p,q}$-orthogonal matrix, hyperbolic QR factorization, bidiagonal factorization, backward stability
AMS subject classification. 65F05, 65F20, 65G50

1 Introduction

We consider the indefinite least squares (ILS) problem

$$\min_x (Ax-b)^T\Sigma_{p,q}(Ax-b), \tag{1}$$

where $A \in \mathbb{R}^{(p+q)\times n}$, $b \in \mathbb{R}^{p+q}$, and $\Sigma_{p,q} = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}$ is a signature matrix. This problem has several applications. Examples include the total least squares problems ([5]) and the H^∞ smoothing problems ([7, 10]). It is known that the ILS problem has a unique solution if and only if

$$A^T\Sigma_{p,q}A > 0, \tag{2}$$

e.g., [7, 5, 2]. In this note we assume that the condition (2) always holds. Note under this condition we have $p \ge n$.

The ILS problem is equivalent to its normal equation

$$A^T\Sigma_{p,q}Ax = A^T\Sigma_{p,q}b. \tag{3}$$

Since the normal equation is usually more ill-conditioned than the ILS problem, numerically one prefers to solve the problem by directly working on the original matrix A and the vector

*Department of Mathematics, University of Kansas, Lawrence, KS 66045, USA. xu@math.ukans.edu. The author was partially supported by National Science Foundation grant 0314427, and the University of Kansas General Research Fund allocation # 2301717.

b. A typical example is the method that uses the QR factorization to solve the standard least squares problem (which is the special case of the ILS with $q = 0$), see, e.g., [1, 9], and [6, Sec. 5.3]. Following the idea of the QR factorization method, recently two methods were developed to solve the general ILS problem. The method proposed in [5] uses the QR factorization of A to solve the ILS problem. The precise procedure is as follows. First compute the compacted QR factorization

$$A = QR,$$

where Q is orthonormal and R is square upper-triangular. Then compute the Cholesky factorization

$$LL^T = Q^T \Sigma_{p,q} Q.$$

Finally compute the solution x by solving three triangular systems successively,

$$Lz = Q^T \Sigma_{p,q} b, \quad L^T y = z, \quad Rx = y.$$

It is easily verified that the solution x satisfies $Q^T \Sigma_{p,q} Q R x = Q^T \Sigma_{p,q} b$. By pre-multiplying R^T, it is just the normal equation (3). In [5] it is shown that this method is numerically backward stable.

The method proposed in [2] uses the hyperbolic QR factorization, an analog of the QR factorization, to solve the ILS problem.

Definition 1 *Let* $\Sigma_{p,q} = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}$

a) The matrix $H \in \mathbb{R}^{(p+q)\times(p+q)}$ *is* $\Sigma_{p,q}$-orthogonal *or* hyperbolic *if* $H^T \Sigma_{p,q} H = \Sigma_{p,q}$.

b) Let $A \in \mathbb{R}^{(p+q)\times n}$. *The factorization*

$$A = H \begin{bmatrix} R \\ 0 \end{bmatrix},$$

is called the hyperbolic QR factorization *of* A *if* H *is* $\Sigma_{p,q}$*-orthogonal and* R *is upper-triangular.*

The method given in [2] consists of the following steps. First compute the hyperbolic factorization

$$A = H \begin{bmatrix} R \\ 0 \end{bmatrix}$$

and simultaneously update the vector

$$g = \begin{bmatrix} I_n & 0 \end{bmatrix} H^T \Sigma_{p,q} b.$$

Then compute x by solving the triangular system

$$Rx = g.$$

This method is very similar to the QR factorization method for the standard least squares problem. Unlike the first method, which still needs to work on the product $Q^T \Sigma_{p,q} Q$, this method directly work on A and b. Therefore it is less expensive (e.g., [2]). In [2] it is also proved that under some mild assumptions the hyperbolic QR factorization method is forward stable. However, it is not clear whether the method is also backward stable. The main problem is that one can only show that the computed hyperbolic QR factorization and vector g satisfy a mixed backward-forward stable error model ([2]).

In this note we combine the ideas that were used for the previous two methods to develop the third method. We will also use the hyperbolic QR factorization. But for numerical stability we will compute the hyperbolic QR factorization of a normalized matrix. The general procedure of the method is given in the following algorithm.

Algorithm 1. *Given* $A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \in \mathbb{R}^{(p+q)\times n}$ *and* $b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \in \mathbb{R}^{p+q}$, *where* $A_1 \in \mathbb{R}^{p\times n}$, $A_2 \in \mathbb{R}^{q\times n}$, $b_1 \in \mathbb{R}^p$, $b_2 \in \mathbb{R}^q$, *and* $A^T\Sigma_{p,q}A > 0$, *the algorithm computes the solution of the indefinite least squares problem (1).*

Step 1. Compute the permuted bidiagonal factorization

$$A_1 = U \begin{bmatrix} 0 \\ D \end{bmatrix} V^T,$$

where U, V are orthogonal and D is upper-bidiagonal.

Compute $d_1 = U^T b_1$.

Step 2. Solve the matrix equation

$$SD = A_2 V$$

for S.

Step 3. Compute

$$f = \begin{bmatrix} 0 & I_n & -S^T \end{bmatrix} \begin{bmatrix} d_1 \\ b_2 \end{bmatrix}.$$

Compute the hyperbolic QR factorization

$$\begin{bmatrix} I_n \\ S \end{bmatrix} = H \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{4}$$

where H is $\Sigma_{n,q}$-orthogonal and R is upper triangular.

Step 4. Compute y by solving the triangular systems successively,

$$R^T w = f, \quad Rz = w, \quad Dy = z.$$

Compute $x = Vy$.

We will discuss the detailed computation process in the next section. In section 3 we will give the first order error analysis and show that the algorithm is numerically backward stable.

For error analysis we will use the standard model of floating point arithmetic ([8, pp. 44]):

$$fl(a \circ b) = (a + b)(1 + \delta), \quad fl(\sqrt{a}) = \sqrt{a}(1 + \delta), \qquad |\delta| \leq \mathbf{u},$$

where $\mathbf{u}$ is the machine precision and $\circ = +, -, \times, /$. The spectral norm for matrices and the 2-norm for vectors are denoted by $\|\cdot\|$. The ith column of the identity matrix I is denoted by e_i.

2 Implementation details

In the algorithm Step 1 and 2 actually compute the factorization

$$A = \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} 0 \\ I_n \\ S \end{bmatrix} DV^T.$$

Step 3 is equivalent to compute the hyperbolic QR factorization of the normalized matrix

$$\begin{bmatrix} 0 \\ I_n \\ S \end{bmatrix} = \begin{bmatrix} I_{p-n} & 0 \\ 0 & H \end{bmatrix} \begin{bmatrix} 0 \\ R \\ 0 \end{bmatrix}.$$

By using these two forms the normal equation (3) becomes

$$VD^T R^T RDV^T x = VD^T \begin{bmatrix} 0 & I_n & S^T \end{bmatrix} \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q} b,$$

which is equivalent to

$$R^T RDV^T x = \begin{bmatrix} 0 & I_n & -S^T \end{bmatrix} \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T b = f.$$

The solution x is then obtained from Step 4.

Note

$$A^T \Sigma_{p,q} A > 0 \Longrightarrow A_1^T A_1 - A_2^T A_2 > 0 \Longrightarrow D^T D - V^T A_2^T A_2 V > 0.$$

So D is nonsingular, and from the last inequality we have $I - S^T S > 0$. This implies that $\|S\| < 1$.

We will discuss Step 1 and Step 3 in details. Other steps are trivial.

In Step 1 the factorization can be obtained by applying the bidiagonal factorization methods followed by a block row permutation. The Householder transformation method for computing the bidiagonal factorization can be found in [6, pp. 252]. Since U is only used for computing d_1, we only need to apply the Householder transformations to b_1 directly during the factorization process without storing U. The matrix V can be stored in the factored form, i.e., only the vectors for the Householder transformations. When $p >> n$ a faster method was proposed in [4]. It consists of two steps. First compute the QR factorization of A_1. Then compute the bidiagonal factorization of the upper-triangular factor.

In Step 3 the hyperbolic QR factorization can be computed by the method as in [2]. But here the top block of the normalized matrix is I_n. So we can use the following simple version. We first introduce the *hyperbolic rotation* matrices

$$G_{ij}(\alpha, \beta) = I_{n+q} + (\alpha - 1)(e_i e_i^T + e_j e_j^T) - \beta(e_i e_j^T + e_j e_i^T),$$

where $1 \le i \le n$, $n+1 \le j \le n+q$ and α, β satisfy $\alpha^2 - \beta^2 = 1$. Clearly $G_{ij}(\alpha,\beta)$ is $\Sigma_{n,q}$-orthogonal. Given a vector $x \in \mathbb{R}^{n+q}$ with $|x_i| > |x_j|$, where the integers i, j satisfy $1 \le i \le n$ and $n+1 \le j \le n+q$, a hyperbolic rotation $G_{ij}(\alpha,\beta)$ can be constructed to zero x_j. The parameters α, β may be chosen as $\alpha = x_i/\sqrt{x_i^2 - x_j^2}$, $\beta = x_j/\sqrt{x_i^2 - x_j^2}$.

We compute the hyperbolic QR factorization (4) by applying the Householder transformations and the hyperbolic rotations to eliminate the entries of $\begin{bmatrix} I_n \\ S \end{bmatrix}$ column by column. The algorithm is given below. In the algorithm we will use the Matlab forms to denote the entries, rows and columns of matrices.

Algorithm for computing the hyperbolic QR factorization of $\begin{bmatrix} I_n \\ S \end{bmatrix}$

Step 0. Set $R = I_n$.

Step 1. For $k = 1 : n$

a) Construct the Householder matrix Q_k such that $Q_k S(:, k) = x_k e_1$,

Compute $S(:, k : n) := Q_k S(:, k : n)$

b) *% Construct the hyperbolic rotation* $G_{k,n+1}(\alpha_k, \beta_k)$ *to eliminate* $S(1, k)$ $(= x_k)$.

% The parameter $\beta_k (= x_k \alpha_k)$ *is not needed.*

Compute $R(k, k) = \sqrt{1 - x_k^2}, \quad \alpha_k = 1/R(k, k)$

c) *% Compute* $\begin{bmatrix} R \\ S \end{bmatrix} := G_{k,n+1}(\alpha_k, \beta_k) \begin{bmatrix} R \\ S \end{bmatrix}$.

$S(1, k + 1 : n) = \alpha_k S(1, k + 1 : n)$

$R(k, k + 1 : n) = -x_k S(1, k + 1 : n)$

End For

Under the condition (2), initially we have $I - S^T S > 0$. Obviously the positive definiteness of $R^T R - S^T S$ is preserved during the reduction process. From this one can verify that $|x_k| < 1$ for all $1 \leq k \leq n$. So the algorithm will not break down.

We discuss the cost of Algorithm 1.

Step 1. It needs about $4pn^2 - 4n^3/3$ flops for the bidiagonal factorization. If the method in [4] is employed, it needs about $2pn^3 + 2n^3$ flops. Computing d_1 needs about $4pn$ flops.

Step 2. It needs about $2qn^2$ flops for computing $A_2 V$, and about $3qn$ flops for solving the bidiagonal system for S.

Step 3 It needs about $2qn$ flops for computing the vector f. It needs about $2qn^2$ flops for computing R. The hyperbolic matrix H doesn't need to be updated.

Step 4. It needs about $2n^2$ flops for solving three systems of equations to get y. Finally computing x needs about $2n^2$ flops.

Table 1 compares the cost of Algorithm 1 with the costs of the methods proposed in [5] and [2]. So about the cost Algorithm 1 is between other two methods.

	Algorithm 1	Algorithm in [5]	Algorithm in [2]
	$(4p+4q-4n/3)n^2$ or $(2p+4q+2n)n^2$	$(5p+5q-n)n^2$	$(2p+2q-2n/3)n^2$
$p >> n$	$(2p+4q)n^2$	$(5p+5q)n^2$	$(2p+2q)n^2$
$p \approx n$	$(8n/3+4q)n^2$	$(4n+5q)n^2$	$(4n/3+2q)n^2$

Table 1: Costs of methods for solving the ILS problem.

3 Error analysis

We will only consider the first order error bounds and ignore the possibility of overflow or underflow. We will use the letters with a hat for the matrices, vectors, or scalars computed in finite arithmetic. To show the backward stability we need the following two auxiliary results.

Lemma 2 *Suppose $D \in \mathbb{R}^{n\times n}$ is nonsingular upper bidiagonal and $B \in \mathbb{R}^{q\times n}$. Let $\hat{X}$ be the numerical solution of the equation*

$$XD = B,$$

computed by forward substitution. Then $\hat{X}$ satisfies

$$(\hat{X} + \Delta X)D = B + \Delta B,$$

where $\|\Delta X\| \le 3n\mathbf{u}\|\hat{X}\|$, $\|\Delta B\| \le 3n\mathbf{u}\|B\|$.

Proof. See Lemma 8 in [3]. □

Lemma 3 *Suppose $R, \Delta R_1, \Delta R_2 \in \mathbb{R}^{n\times n}$, and $\|\Delta R_1\|, \|\Delta R_2\| = O(\mathbf{u}\|R\|)$. Then the vector $y = (R+\Delta R_1)^T(R+\Delta R_2)x$ can be expressed as*

$$y = (R^TR + \Delta R_3)x,$$

where ΔR_3 is symmetric and $\|\Delta R_3\| \le O(\mathbf{u}\|R\|^2)$.

Proof. See [5]. □

The bidiagonal matrix $\hat{D}$ computed in Step 1 satisfies

$$A_1 + \Delta A_1 = U \begin{bmatrix} 0 \\ \hat{D} \end{bmatrix} V^T, \tag{5}$$

where U, V are orthogonal and $\|\Delta A_1\| = 0(\mathbf{u}\|A_1\|)$, (see, e.g.,[6, Sec. 5.5]).

The computed vector $\hat{d}_1$ satisfies

$$\hat{d}_1 = U^T(b_1 + \Delta b_1), \tag{6}$$

where U is orthogonal, same as that in (5), and $\|\Delta b_1\| = O(\mathbf{u}\|b_1\|)$, ([8, Lemma 18.3]). Based on the same error analysis for the product A_2V and using Lemma 2, the matrix $\hat{S}$ computed in Step 2 satisfies

$$(\hat{S} + \Delta S_1)\hat{D} = (A_2 + \Delta A_2)V, \tag{7}$$

where V is orthogonal, same as that in (5), $\|\Delta S_1\| = O(\mathbf{u}\|\hat{S}\|)$, $\|\Delta A_2\| = O(\mathbf{u}\|A_2\|)$.

In Step 3, the computed factor $\hat{R}$ in the hyperbolic factorization satisfies ([2])

$$\begin{bmatrix} \hat{R} + \Delta R_1 \\ \hat{S} + \Delta S_2 \end{bmatrix} = Q \begin{bmatrix} I + \Delta E_1 \\ 0 \end{bmatrix}, \tag{8}$$

where Q is orthogonal (which is related to H), $\|\Delta R_1\|, \|\Delta S_2\| = O(\mathbf{u} \max\{\|\hat{R}\|, \|\hat{S}\|\})$, $\|\Delta E_1\| = O(\mathbf{u})$.

The computed vector $\hat{f}$ satisfies

$$\hat{f} = \begin{bmatrix} 0 & I_n \end{bmatrix} (\hat{d}_1 + \Delta d_1) - (\hat{S} + \Delta S_3) b_2, \tag{9}$$

where $\|\Delta d_1\| = O(\mathbf{u}\|\hat{d}_1\|) = O(\mathbf{u}\|b_1\|)$, $\|\Delta S_3\| = O(\mathbf{u}\|\hat{S}\|)$ (see [8, pp. 76]). The vectors computed in Step 4 satisfy

$$\begin{aligned} (\hat{R} + \Delta R_2)^T \hat{w} &= \hat{f} & (10)\\ (\hat{R} + \Delta R_3)\hat{z} &= \hat{w} & (11)\\ (\hat{D} + \Delta D)\hat{y} &= \hat{z}, & (12) \end{aligned}$$

where $\|\Delta R_2\|, \|\Delta R_3\| \leq n\mathbf{u}\|\hat{R}\|$, $\|\Delta D\| \leq 3\mathbf{u}\|\hat{D}\|$, see, e.g., [8, Sec. 8.1].

Finally the computed solution $\hat{x}$ satisfies

$$\hat{x} = (V + \Delta V)\hat{y}, \tag{13}$$

where $\|\Delta V\| = O(\mathbf{u})$, (see [8, Lemma 18.2]).

By using the formulas (10) – (13), and (6), (9), we have

$$\begin{aligned} \hat{f} &= (\hat{R} + \Delta R_2)^T (\hat{R} + \Delta R_3)(\hat{D} + \Delta D)(V + \Delta V)^T \hat{x} \\ &= \begin{bmatrix} 0 \\ I_n \\ \hat{S} + \Delta S_3 \end{bmatrix}^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b_2), \end{aligned} \tag{14}$$

where $\Delta b_2 = \begin{bmatrix} \Delta b_1 + U\Delta d_1 \\ 0 \end{bmatrix}$. So

$$\|\Delta b_2\| = O(\mathbf{u}\|b_1\|). \tag{15}$$

Taking $(\hat{D} + \Delta D)(V + \Delta V)^T \hat{x}$ as a single vector, by Lemma 3 we have

$$(\hat{R} + \Delta R_2)^T (\hat{R} + \Delta R_3)(\hat{D} + \Delta D)(V + \Delta V)^T \hat{x} = (\hat{R}^T \hat{R} + \Delta R_4)(\hat{D} + \Delta D)(V + \Delta V)^T \hat{x}$$

where $\Delta R_4 = (\Delta R_4)^T$ and $\|\Delta R_4\| = O(\mathbf{u}\|\hat{R}\|^2)$.

From (8) we have

$$(\hat{R} + \Delta R_1)^T (\hat{R} + \Delta R_1) + (\hat{S} + \Delta S_2)^T (\hat{S} + \Delta S_2) = (I_n + \Delta E_1)^T (I_n + \Delta E_1). \tag{16}$$

It can be rewritten as

$$\hat{R}^T \hat{R} = I_n - (\hat{S} + \Delta S_1)^T (\hat{S} + \Delta S_1) + \Delta R_5,$$

where ΔS_1 is defined in (7)

$$\Delta R_5 = (\Delta E_1)^T + \Delta E_1 - \hat{R}^T \Delta R_1 - (\Delta R_1)^T \hat{R} - \hat{S}^T(\Delta S_2 - \Delta S_1) - (\Delta S_2 - \Delta S_1)^T \hat{S} + O(\mathbf{u}^2)$$

is symmetric. Then

$$\hat{R}^T \hat{R} + \Delta R_4 = I_n + \Delta R_4 + \Delta R_5 - (\hat{S} + \Delta S_1)^T (\hat{S} + \Delta S_1).$$

Note (16) also implies that $\|\hat{R}\|, \|\hat{S}\| \leq 1 + O(\mathbf{u})$. So $\|\Delta R_4 + \Delta R_5\| = O(\mathbf{u})$. If

$$\|\Delta R_4 + \Delta R_5\| < 1, \tag{17}$$

the matrix $I_n + \Delta R_4 + \Delta R_5$ is symmetric positive definite. In this case it is well known that $I_n + \Delta R_4 + \Delta R_5$ has a unique principle square root, i.e., there exists a symmetric matrix ΔE_2 such that

$$(I_n + \Delta E_2)^2 = I_n + \Delta R_4 + \Delta R_5,$$

and $\|\Delta E_2\| \leq \frac{1}{2}\|\Delta R_4 + \Delta R_5\| = O(\mathbf{u})$. With this form we have

$$\hat{R}^T \hat{R} + \Delta R_4 = (I_n + \Delta E_2)^2 - (\hat{S} + \Delta S_1)^T (\hat{S} + \Delta S_1) =: \tilde{F}^T \Sigma_{p,q} \tilde{F},$$

where

$$\tilde{F} = \begin{bmatrix} 0 \\ I_n + \Delta E_2 \\ \hat{S} + \Delta S_1 \end{bmatrix}.$$

The first equation in (14) now becomes

$$\hat{f} = (\tilde{F}^T \Sigma_{p,q} \tilde{F})(\hat{D} + \Delta D)(V + \Delta V)^T \hat{x}. \tag{18}$$

The matrix

$$\tilde{F}^T \tilde{F} = (I_n + \Delta E_2)^2 + (\hat{S} + \Delta S_1)^T (\hat{S} + \Delta S_1)$$

is positive definite. When (17) holds, we have

$$\|(\tilde{F}^T \tilde{F})^{-1}\| \leq \|(I + \Delta E_2)^{-2}\| = 1 + O(\mathbf{u}). \tag{19}$$

Denote

$$\Delta F = \begin{bmatrix} 0 \\ \Delta E_2 \\ \Delta S_1 - \Delta S_3 \end{bmatrix}.$$

Obviously $\|\Delta F\| = O(\mathbf{u})$, and

$$\begin{bmatrix} 0 \\ I_n \\ \hat{S} + \Delta S_3 \end{bmatrix} = \tilde{F} - \Delta F.$$

Applying the same trick used in [5] to the right-hand side vector in (14),

$$\hat{f} \quad = \quad (\tilde{F} - \Delta F)^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b_2)$$

$$= (\tilde{F} - \Delta F(\tilde{F}^T\tilde{F})^{-1}\tilde{F}^T\tilde{F})^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b_2)$$

$$= \tilde{F}^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q} \left(\Sigma_{p,q} \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \left(I_{p+q} - \tilde{F}(\tilde{F}^T\tilde{F})^{-1}(\Delta F)^T \right) \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b_2) \right)$$

$$= \tilde{F}^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q} \left(b + \Delta b_2 - \Sigma_{p,q} \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \tilde{F}(\tilde{F}^T\tilde{F})^{-1}(\Delta F)^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q} b + O(\mathbf{u}^2) \right).$$

Let

$$\Delta b = \Delta b_2 - \Sigma_{p,q} \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \tilde{F}(\tilde{F}^T\tilde{F})^{-1}(\Delta F)^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q} b + O(\mathbf{u}^2).$$

Then by (19),

$$\|\tilde{F}(\tilde{F}^T\tilde{F})^{-1}\| = \sqrt{\|(\tilde{F}^T\tilde{F})^{-1}\tilde{F}^T\tilde{F}(\tilde{F}^T\tilde{F})^{-1}\|} = \sqrt{\|(\tilde{F}^T\tilde{F})^{-1}\|} \le 1 + O(\mathbf{u}).$$

So by (15) and the fact that $\|\Delta F\| = O(\mathbf{u})$, we have

$$\|\Delta b\| \le \|\Delta b_2\| + \|b\|\|\Delta F\| = O(\mathbf{u}\|b\|).$$

Now we have

$$\hat{f} = \tilde{F}^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b). \tag{20}$$

By (18) and (20) we have

$$(\tilde{F}^T\Sigma_{p,q}\tilde{F})(\hat{D} + \Delta D)(V + \Delta V)^T\hat{x} = \tilde{F}^T \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix}^T \Sigma_{p,q}(b + \Delta b).$$

Let

$$\tilde{A} = \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \tilde{F}(\hat{D} + \Delta D)(V + \Delta V)^T.$$

Then by pre-multiplying $(V + \Delta V)(\hat{D} + \Delta D)^T$ to the above equation, we have

$$\tilde{A}^T\Sigma_{p,q}\tilde{A}\hat{x} = \tilde{A}\Sigma_{p,q}(b + \Delta b).$$

By using (5) and (7)

$$\Delta A := \tilde{A} - A = \begin{bmatrix} \Delta A_1 \\ \Delta A_2 \end{bmatrix} + \begin{bmatrix} U & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} 0 \\ \Delta E_2\hat{D}V^T + \hat{D}(\Delta V)^T + \Delta D V^T \\ \hat{S}\Delta D V^T + A_2 V(\Delta V)^T \end{bmatrix} + O(\mathbf{u}^2).$$

Since $\|\hat{D}\| = \|A_1\| + O(\mathbf{u}\|A_1\|)$, $\|\Delta D\| = O(\mathbf{u}\|\hat{D}\|)$, $\|\hat{S}\| \le 1 + O(\mathbf{u})$, and $\|\Delta E_2\|, \|\Delta V\| = O(\mathbf{u})$, we have

$$\|\Delta A\| = O(\mathbf{u}\|A\|).$$

Therefore the solution $\hat{x}$ computed by Algorithm 1 satisfies

$$(A + \Delta A)^T\Sigma_{p,q}(A + \Delta A)\hat{x} = (A + \Delta A)^T\Sigma_{p,q}(b + \Delta b),$$

or equivalently, $\hat{x}$ solves the perturbed ILS problem

$$\min_x \left((A + \Delta A)x - (b + \Delta b)\right)^T \Sigma_{p,q} \left((A + \Delta A)x - (b + \Delta b)\right)^T,$$

where $\|\Delta A\| = O(\mathbf{u}\|A\|)$ and $\|\Delta b\| = O(\mathbf{u}\|b\|)$. So Algorithm 1 is backward stable.

4 Conclusion

We proposed a numerically backward stable method for solving the indefinite least squares problem. The method employs the hyperbolic QR factorization. It is more efficient than the backward stable method based on the QR and Cholesky factorizations. It is known that in general the hyperbolic QR factorization methods are mixed stable. But for the ILS problem by carefully implementing such a method, a backward stable algorithm still can be constructed.

References

[1] Å. Björck, *Numerical Method for Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.

[2] A. Bojanczyk, N.J. Higham, and H. Patel, *Solving the indefinite least squares problem by hyperbolic QR factorization,* SIAM J. Matrix Anal. Appl., 24 (2003), pp. 914-931.

[3] R. Byers and H. Xu, *A rapidly converging, backward stable matrix sign function method for computing polar decomposition,* In progress.

[4] T.F. Chan, *An improved algorithm for computing the singular value decomposition,* ACM Trans. Math. Soft. 8 (1982), pp. 72-83.

[5] S. Chandrasekaran, M. Gu, and A.H. Sayed, *A stable and efficient algorithm for the indefinite linear least-squares problem,* SIAM J. Matrix Anal. Appl., 20 (1998), pp. 354-362.

[6] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third edition, 1996.

[7] B. Hassibi, A.H. Sayed, and T. Kailath, *Linear estimation in Krein spaces - Part I: Theory*, IEEE Trans. Automat. Control, 41 (1996), pp. 18-33.

[8] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.

[9] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1995.

[10] A.H. Sayed, B. Hassibi, and T. Kailath, *Inertia properties of indefinite quadratic forms*, IEEE Signal Process. Lett., 3 (1996), pp. 57-59.