Self-Scaling Fast Rotations for Stiff and Equality Constrained Linear Least Squares Problems

Andrew A. Anda * and Haesun Park †

Abstract

We present algorithms which apply self-scaling fast plane rotations to the QR decomposition for stiff least squares problems. We show that both fast and standard Givens rotation-based algorithms produce accurate results regardless of row sorting and even with extremely large weights when equality constrained least squares problems are solved by the weighting method. Numerical test results show that the Householder QR decomposition algorithm is sensitive to row sorting and produces less accurate results when the weights are large and the modified Gram-Schmidt algorithm is less sensitive to row sorting. This makes the fast plane rotation a method of choice for the QR decomposition of stiff matrices since it is also competitive in computational complexity. Based on the above results, an efficient algorithm is also derived for the application where the least squares solutions are required for various constrained matrices for each fixed data matrix.

Keywords: equality constrained least squares, Givens rotation, Householder transformation, fast plane rotation, stiff least squares, weighted least squares

---

*Computer Science Department, University of Minnesota, Minneapolis, MN 55455. The work of both authors was supported by a contract DAAL02-89-C-0038 between the Army Research Office and the University of Minnesota for the Army High Performance Computing Research Center.

†Computer Science Department, University of Minnesota, Minneapolis, MN 55455. The work of this author was also supported in part by National Science Foundation grant CCR-9209726.

1 Introduction

Least squares problems with equality constraints (LSE) may be represented as

$$\min_{Bx=d} \|Ax - b\|_2 , \tag{1}$$

where $A \in \mathbf{R}^{m\times n}, B \in \mathbf{R}^{l\times n}, m \geq n,$ and $n \geq l$. We will assume that $\text{rank}(B) = l$, and $\text{null}(A) \bigcap \text{null}(B) = \{0\}$. If $\text{rank}\left(\begin{bmatrix} B \\ A \end{bmatrix}\right) < n$, a minimum norm solution can be specified.

The LSE problems arise in several applications, including adaptive beamforming in signal processing [VL85], curve fitting [Bjö90], penalty function methods in nonlinear optimization [VL83], geodetic least squares adjustment [VL85], and surface fitting [LH74].

The methods for solving the LSE problem include the *nullspace method*, the *direct elimination method*, and the *weighting method* [LH74]. All of these methods involve orthogonal transformations. As methods based on the solution of normal equations perform much worse when matrix condition numbers are large, they are not generally recommended. The LSE problem may also be solved via pseudo-inverses (and thus via the singular value decomposition (SVD)) [LH74], but because of its greater computational complexity, this method is of practical use for analysis only. For the analyses based on the generalized SVD (GSVD) and on weighted pseudo-inverses, see [Bar88, Eld80, Eld82, VL85]. In [Bjö90], it has been illustrated that the nullspace and the direct elimination methods are numerically stable and these two methods are shown to yield almost identical numerical accuracy results [LW70]. There are also direct and iterative methods for the solution of sparse problems which are beyond the scope of this article. Readers are directed to [BNP88].

We have previously presented self-scaling fast plane rotations [AP94] which obviate the rescaling necessary in other fast plane rotations. In this paper, we present algorithms that apply self-scaling fast plane rotations to the QR decomposition for stiff least squares problems. These problems appear when an equality constrained linear least square problem is solved via extreme weighting of the constraint equations, for example. The accuracy of our algorithm compares favorably with that of the Givens rotation based algorithm while the Householder method may produce very sensitive results. Moreover, both fast and standard Givens rotation based algorithms produce very accurate results regardless of row sorting and even with extremely large weights, in our experiments. This makes the fast plane rotation a method of choice for the QR decomposition since it is also competitive in complexity with the Householder method.

This paper is organized as follows: In Section 2, we review the methods for solving equality constrained least squares problems and compare their computational complexities. In Section 3, we describe how the self-scaling fast rotation can be applied in solving equality constrained least squares problems via the weighting method. The role of row sorting and the column pivoting in producing accurate results in the weighting method is discussed. We show how the presented algorithm can be utilized in applications where the least squares solutions are required for various constrained matrices for each fixed data matrix. We then present the numerical test results in Section 4, the Givens methods are shown to produce far more accurate results than the Householder method and the modified Gram-Schmidt method in extreme cases when the weight values become large and rows are sorted in a certain way.

2 Equality Constrained Least Squares

In this section, we briefly review weighting method, null space method, and the direct elimination method and compare their computational complexities. In the weighting method, the LSE problem (1) is transformed to the unconstrained linear least squares (LS) problem,

$$\min_x \left\| \begin{bmatrix} \eta B \\ A \end{bmatrix} x - \begin{bmatrix} \eta d \\ b \end{bmatrix} \right\|_2, \ \eta \gg 1. \tag{2}$$

Which is the same as the weighted least squares problem (WLS) [Bjö90]

$$\min_x \left\| W \left(\begin{bmatrix} B \\ A \end{bmatrix} x - \begin{bmatrix} d \\ b \end{bmatrix} \right) \right\|_2, \tag{3}$$

where

$$W = diag(\eta I_l, \ I_{m-l}). \tag{4}$$

It is shown [GVL89] that with a large enough η, the solution to (2) can accurately approximate the LSE solution to (1). However, a large value of η yields a *stiff* problem and we have to choose the algorithm carefully for accurate results.

The normal equation method of solution

$$(\eta^2 B^T B + A^T A)x = \eta^2 B^T d + A^T b \tag{5}$$

should be avoided when the $\eta^2 B^T B$ term overwhelms the $A^T A$ term resulting an unacceptable information loss for large values of η.

The Householder, modified Gram-Schmidt, and Givens orthogonal decomposition methods are commonly used for the solution of the LS problems [Bjö91, GVL89]. It has been shown that for the stiff problems, the solution vector obtained by the QR decomposition [1] is sensitive to the row sorting of the matrix and also to the size of η. However, in this paper, we will show that although this is the case with the Householder method, *the Givens method and its fast versions for the QR decomposition are not sensitive to row sorting* according to a substantial number of experiments.

The nullspace method [LH74], summarized in Algorithm 1, uses an orthogonal basis of the null space of the constraint matrix. Although this method admits stable updating, it is inefficient if A is large and sparse [GVL89] or if the problem is to be solved for various B matrices for a fixed A, as each product, AQ_B, must be recalculated.

ALGORITHM 1 (NULLSPACE).

$$\begin{aligned}
\begin{bmatrix} B \\ A \end{bmatrix} Q_B &\Longrightarrow \begin{bmatrix} L_B & 0 \\ \tilde{A}_1 & \tilde{A}_2 \end{bmatrix},\ Q_B^T Q_B = I,\ L_B \in \mathbb{R}^{l\times l}\ \textit{(LQ transform)} \\
L_B^{-1} d &\Longrightarrow \bar{x}_B\ \textit{(triangular forward solve)} \\
b - \tilde{A}_1 \bar{x}_B &\Longrightarrow \tilde{b} \\
\min_{\bar{x}_A} \left\| \tilde{A}_2 \bar{x}_A - \tilde{b} \right\|_2 &\Longrightarrow \bar{x}_A \in \mathbb{R}^{(n-l)} \textit{(LS solve)} \\
Q_B \begin{bmatrix} \bar{x}_B \\ \bar{x}_A \end{bmatrix} &\Longrightarrow x \qquad \Box
\end{aligned}$$

The direct elimination method is presented in Algorithm 2. It applies orthogonal transformations to the constraint matrix and then elementary transformations

[1] As an important implementation detail, if matrices are stored in row-major form, the $X = QR$ decomposition should be performed, else, if storage is in column-major form, the $X^T = LQ^T$ decomposition should be performed so as to promote and maximize contiguous data reference. Herein we will assume row-major storage and discuss the QR rather than the LQ^T decomposition.

TABLE 1
Computational complexity of the LSE solvers

weighting	null space	direct elimination
$\frac{1}{2}\nu n^2\left(m+l-\frac{n}{3}\right)$	$\frac{1}{2}\nu n^2\left(m+l-\frac{n}{3}\right)$	$\frac{1}{2}\nu n^2\left(m+l-\frac{n}{3}\right)-mnl\,(\nu-2)$

ν = operations for a 2 element rotation. $\nu=6$ flops (slow rotation), $\nu=4$ flops (fast rotation)

(Gaussian elimination) to the data matrix. Column interchanges are necessary to insure that the resulting first l columns of the constraint matrix are linearly independent. Then, the equality constrained least squares problem (1), can be restated as the unconstrained problem. For the generalization of this algorithm to handle the instance where A and B have certain sparsity structure and each may be rank deficient, see [Bjö84]. The direct elimination method admits updating in a straightforward manner.

ALGORITHM 2 (DIRECT ELIMINATION).

$$\begin{bmatrix} Q_B^T & 0 \\ 0 & I_m \end{bmatrix}\begin{bmatrix} B & d \\ A & b \end{bmatrix}\Pi \implies Q_B^T\begin{bmatrix} B_1 & B_2 & d \\ A_1 & A_2 & b \end{bmatrix} = \begin{bmatrix} R_B & \tilde{B}_2 & \tilde{d} \\ A_1 & A_2 & b \end{bmatrix},$$

$$B_1 \in \mathbb{R}^{l\times l}, R_B \in \mathbb{R}^{l\times l} \ \textit{(QR with column interchange)}$$

$$\begin{bmatrix} I_l & 0 \\ M & I_m \end{bmatrix}\begin{bmatrix} R_B & \tilde{B}_2 & \tilde{d} \\ A_1 & A_2 & b \end{bmatrix} \implies \begin{bmatrix} R_B & \tilde{B}_2 & \tilde{d} \\ 0 & \tilde{A}_2 & \tilde{b} \end{bmatrix} \ \textit{(Gaussian elim.)}$$

$$\begin{bmatrix} I_l & 0 \\ 0 & Q_A^T \end{bmatrix}\begin{bmatrix} R_B & \tilde{B}_2 & \tilde{d} \\ 0 & \tilde{A}_2 & \tilde{b} \end{bmatrix} \implies \begin{bmatrix} R_B & \tilde{B}_2 & \tilde{d} \\ 0 & R_A & \hat{b}_1 \\ 0 & 0 & \hat{b}_2 \end{bmatrix},\ R_A \in \mathbb{R}^{(n-l)\times(n-l)}$$

$$\begin{bmatrix} R_B & \tilde{B}_2 \\ 0 & R_A \end{bmatrix}^{-1}\begin{bmatrix} \tilde{d} \\ \hat{b}_1 \end{bmatrix} \implies \bar{x} \ \textit{(triangular backsolve)}$$

$$\Pi\bar{x} \implies x \qquad \square$$

In Table 1, the floating point complexity (1 flop $\approx$ 1 multiplication or 1 addition) is presented using the efficient algorithm for each method. It was assumed that the orthogonal factor was not formed explicitly but the rotations which would form it were saved in factored form and later used as described in [Ste76].

Because the null space and weighting methods consist of rotations and triangular solves exclusively, their floating point complexities are roughly the same. The direct elimination method replaces some orthogonal transformations with elementary transformations which are computationally less expensive. However, this may result in less accurate results, e.g. when the elements in A are much larger than those in B.

3 Fast Givens Rotations for Stiff LS Problems

3.1 Fast Givens Rotations

The Givens algorithm for the QR decomposition is based on plane rotations. A plane rotation G of order n through an angle θ in a (p,q) plane is the same as the identity matrix I_n, except for the four elements at the intersections of the p-th and q-th rows and columns. It is well known that a Givens rotation can annihilate a specific element in a matrix as

$$(6) \quad \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_{p1} \\ x_{q1} \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}, \quad c = x_{p1}/\sqrt{x_{p1}^2 + x_{q1}^2} \text{ and } \quad s = x_{q1}/\sqrt{x_{p1}^2 + x_{q1}^2}.$$

(In practice, these equations are reformulated to minimize overflow and roundoff error [GVL89].) There are many orderings in which the elements in the matrix can be annihilated for the triangularization. Some of these orderings promote the data locality and others permit many rows to be annihilated in parallel [AP].

For two row vectors of length n, applying each Givens rotation requires $4n$ multiplications (and $2n$ additions). This complexity in multiplication can be reduced to $2n$ using *fast* Givens rotations [Gen73, Ham74, AP94] . A secondary advantage of the fast rotation is that the square roots for the computation of cosine and sine can be eliminated [Rat82]. However, the standard fast rotations have been avoided in production algorithms which utilize plane rotations, mainly due to their possible overflow/underflow problems [BI87, GVL89, LHKK79a, LHKK79b]. We now briefly review our self-scaling fast rotations [AP94], which obviate the need for detecting the overflow/underflow, to be used to solve the WLS (3).

Suppose a transformation via a rotation G gives

$$X' = GX.$$

In fast rotations, the number of multiplications is reduced by keeping the matrix X

in the factored form DY, where D is a diagonal matrix and Y is accordingly scaled, and these two factors are updated separately. The calculation of the product of the two factors may be postponed until the explicit result is required. The matrix D can be initialized as the identity matrix or a diagonal matrix with its diagonal elements the same as the weights. The rotation may then be represented in the factored form

$$X' = GX = GDY = D'FY = D'Y'. \tag{7}$$

In actual computation, in order to avoid square roots, D^2 rather than D is stored and used for the calculation of rotation parameters. There are several ways to choose a fast rotation F and the new diagonal matrix D'^2 so that the number of multiplications is reduced by half compared to the standard (slow) rotation. For the choice of F and D' in the standard fast rotations, see Table 2, where F_{pq} denotes the 2×2 submatrix of F in the (p, q) plane and it is assume that the rotation occurs in the (p, q) plane.

To bound the maximum decrease in the diagonal factor matrix D^2, one must choose between the two alternative formulations of the fast rotations which update the diagonal elements of D^2 with cosines or sines. In the standard fast rotation, although the decrease in magnitude of each element of the diagonal factor D^2 can be bounded by $1/2$ with the appropriate application of the two formulations, the diagonal elements of D^2 are reduced at each rotation and may eventually cause underflow.

The result of applying a standard fast rotation in (7), $Y' = FY$, can be written in a *modified* form

$$\left\{ \begin{array}{lcl} y'_p & \Leftarrow & y_p + t\frac{d_q}{d_p} y_q \\ y'_q & \Leftarrow & y_q - cs\frac{d_p}{d_q} y'_p \\ & & \\ {d^2}'_p & \Leftarrow & c^2 d_p^2 \\ {d^2}'_q & \Leftarrow & c^{-2} d_q^2 \end{array} \right\}. \tag{8}$$

An extra benefit of this formulation, in addition to the elimination of temporary storage of y_p [dR89], is that the value c^2 is multiplied into one diagonal element and divided into the other. Subsequently, the decrease of the diagonal elements is no longer monotonic. In [AP94], we developed four variations of the modified fast rotation (8) for the cases of large and small rotation angles and the cases when the ordering of the p and q indices is reversed. These four rotations were combined into an algorithm which effects a minimization of the drift away from unity of the diagonal factor elements. The decision table for the self-scaling fast plane rotation is also displayed in Table 2. In the table, r denotes the value that satisfies

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} d_p y_{p1} \\ d_q y_{q1} \end{bmatrix} = \begin{bmatrix} d'_p & 0 \\ 0 & d'_q \end{bmatrix} \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

The underlying heuristic for this self-scaling algorithm is to diminish the larger diagonal element while augmenting the lesser. Thus the monitoring and often necessary rescaling of the system, as implemented in [LHKK79a, LHKK79b], is obviated. Additional advantage of the dual AXPY chaining and the elimination of the temporary store should yield performance gains on more advanced architectures [AP].

3.2 Row Pivoting in Self-Scaling Fast Rotations

In this section, we present the error analysis for the self-scaling square-root free fast rotations and discuss the role of row sorting in producing accurate results. The following analysis is based on the error analysis of the standard fast Givens rotation as derived by Parlett [Par80]. We assume that we are at some intermediate stage in the QR decomposition by self-scaling fast rotations. Let ϵ, which may be different at every instance, be a tiny number such that $|\epsilon| \leq \mathbf{u}$ where $\mathbf{u}$ is the *unit roundoff* of the architecture, and let it represent the standard error of an elementary floating-point operation. We will use the notational shorthand of representing $\prod_1^k(1+\epsilon)$ as $(1+k\epsilon)$ and a computed quantity will be differentiated from its exact value by the following convention: $\mathrm{fl}(\alpha) = \hat{\alpha}$. An updated quantity will be represented with a *prime* affixed.

We now examine the small angle case $|\theta| \leq \frac{\pi}{4}$, with $|d_p| \geq |d_q|$, and the large angle case $\frac{\pi}{2} > |\theta| > \frac{\pi}{4}$, with $|d_p| < |d_q|$. The small angle case arises when $|x_{p1}| > |x_{q1}|$. This will usually be the case when x_{p1} is an element of the constraint matrix B but x_{q1} is not, in the weighting method since the constraint is heavily weighted.

The scalars computed in floating point arithmetic are shown in Table 3 in the same order they appear in Table 2. As an improvement over previous analyses [AP94], we introduce the new error variable ϵ_δ to reflect the different nature of that operation's error behavior and to yield a tighter error bound. Note that for $|\theta| \leq \frac{\pi}{4}$, if $|\theta|$ diminishes, $\tau\beta = t^2$ correspondingly diminishes, and thus the more least-significant bits will be shifted away in the process of normalization when added

TABLE 2

Fast plane rotation algorithms with conditions of applicability

	Standard	**Self-Scaling**	
		$\lvert d_p\rvert \geq \lvert d_q\rvert$	$\lvert d_p\rvert < \lvert d_q\rvert$
$\lvert\theta\rvert \leq \frac{\pi}{4}$ $\equiv$ $\frac{y_{q1}^2}{y_{p1}^2} \leq \frac{d_p^2}{d_q^2}$	$\alpha \Leftarrow y_{q1}/y_{p1}$ $\gamma = d_p^2/d_q^2$ $\beta \Leftarrow \alpha/\gamma$ $\delta = 1+\beta*\alpha$ $F_{pq} = \begin{bmatrix} 1 & \beta \\ -\alpha & 1 \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_p^2/\delta \\ d^{2\prime}_q \Leftarrow d_q^2/\delta \end{array} \right\}$ $r \Leftarrow y_{p1}*\delta$	$\gamma = d_p^2/d_q^2$ $\tau = y_{q1}/y_{p1}$ $\beta \Leftarrow \tau/\gamma$ $\delta = 1+\beta*\tau$ $\alpha \Leftarrow \tau/\delta$ $F_{pq} = \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \end{bmatrix} \begin{bmatrix} 1 & \beta \\ 0 & 1 \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_p^2/\delta \\ d^{2\prime}_q \Leftarrow d_q^2*\delta \end{array} \right\}$ $r \Leftarrow y_{p1}*\delta$	$\alpha \Leftarrow y_{q1}/y_{p1}$ $\gamma = d_p^2/d_q^2$ $\tau = \alpha/\gamma$ $\delta = 1+\alpha*\tau$ $\beta \Leftarrow \tau/\delta$ $F_{pq} = \begin{bmatrix} 1 & \beta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\alpha & 1 \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_p^2*\delta \\ d^{2\prime}_q \Leftarrow d_q^2/\delta \end{array} \right\}$ $r \Leftarrow y_{p1}$
$\lvert\theta\rvert > \frac{\pi}{4}$ $\equiv$ $\frac{y_{q1}^2}{y_{p1}^2} > \frac{d_p^2}{d_q^2}$	$\alpha \Leftarrow y_{p1}/y_{q1}$ $\gamma = d_p^2/d_q^2$ $\beta \Leftarrow \alpha*\gamma$ $\delta = 1+\beta*\alpha$ $F_{pq} = \begin{bmatrix} \beta & 1 \\ -1 & \alpha \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_q^2/\delta \\ d^{2\prime}_q \Leftarrow d_p^2/\delta \end{array} \right\}$ $r \Leftarrow y_{q1}*\delta$	$\alpha \Leftarrow y_{p1}/y_{q1}$ $\gamma = d_p^2/d_q^2$ $\tau = \alpha*\gamma$ $\delta = 1+\alpha*\tau$ $\beta \Leftarrow \tau/\delta$ $F_{pq} = \begin{bmatrix} 1 & -\beta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & \alpha \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_q^2*\delta \\ d^{2\prime}_q \Leftarrow d_p^2/\delta \end{array} \right\}$ $r \Leftarrow y_{q1}$	$\gamma = d_p^2/d_q^2$ $\tau = y_{p1}/y_{q1}$ $\beta \Leftarrow \tau*\gamma$ $\delta = 1+\beta*\tau$ $\alpha \Leftarrow \tau/\delta$ $F_{pq} = \begin{bmatrix} 1 & 0 \\ \alpha & -1 \end{bmatrix} \begin{bmatrix} \beta & 1 \\ 1 & 0 \end{bmatrix}$ $\left\{ \begin{array}{l} d^{2\prime}_p \Leftarrow d_q^2/\delta \\ d^{2\prime}_q \Leftarrow d_p^2*\delta \end{array} \right\}$ $r \Leftarrow y_{q1}*\delta$

"$=$" denotes an intermediate value assignment, and "$\Leftarrow$" denotes a final value assignment.

Note: the squares of the diagonal elements are implicit.

to 1. The possible error in $\tau\beta$ will be exhibited in the least-significant bits, so smaller angles correspond to equal or smaller maximum errors in δ, $|\theta_1| < |\theta_2| \Rightarrow \max|\hat{\delta}(\theta_1) - \delta(\theta_1)| \leq \max|\hat{\delta}(\theta_2) - \delta(\theta_2)|$. This relation may not hold if $|\theta_1|$ is too close to $|\theta_2|$. It should be noted though that even if $|(\hat{\tau}\hat{\beta} - \tau\beta)/\tau\beta| \approx \mathbf{u}$, and $|(\hat{\delta} - \delta)/\delta| \approx \mathbf{u}$, it is possible, for $|\theta| \ll \frac{\pi}{4}$, that $|((\hat{\delta} - 1) - \tau\beta)/\tau\beta| \gg \mathbf{u}$. This is

TABLE 3

Self-scaling fast rotation scalars floating point intermediate errors

$\lvert\theta\rvert \le \frac{\pi}{4},\ \lvert d_p\rvert \ge \lvert d_q\rvert$	$\frac{\pi}{2} > \lvert\theta\rvert > \frac{\pi}{4},\ \lvert d_p\rvert < \lvert d_q\rvert$
$\hat{\gamma} = \frac{d_p^2}{d_q^2}(1+\epsilon)$	$\hat{\gamma} = \frac{d_p^2}{d_q^2}(1+\epsilon)$
$\hat{\tau} = \frac{y_{q1}}{y_{p1}}(1+\epsilon) \quad \left\{= t\frac{d_p}{d_q}\right\}$	$\hat{\tau} = \frac{y_{p1}}{y_{q1}}(1+\epsilon) \quad \{=(t\gamma)^{-1}\}$
$\hat{\beta} = \frac{\tau}{\gamma}(1+3\epsilon) \quad \left\{= \frac{d_q^2}{d_p^2}\frac{y_{q1}}{y_{p1}} = t\frac{d_q}{d_p}\right\}$	$\hat{\beta} = \tau\gamma(1+3\epsilon) \quad \left\{= \frac{\gamma}{t}\right\}$
$\hat{\delta} = (1+\tau\beta(1+5\epsilon))(1+\epsilon) \quad \{= c^{-2}\}$	$\hat{\delta} = (1+\tau\beta(1+5\epsilon))(1+\epsilon) \quad \{= s^{-2}\}$
$= (1+\tau\beta)(1+\epsilon_\delta) \quad \{1 \ge \delta \ge 2\}$	$= (1+\tau\beta)(1+\epsilon_\delta) \quad \{1 \ge \delta \ge 2\}$
$\hat{\alpha} = \frac{\tau}{\delta}(1+2\epsilon+\epsilon_\delta) \quad \left\{= sc\frac{d_p}{d_q}\right\}$	$\hat{\alpha} = \frac{\tau}{\delta}(1+2\epsilon+\epsilon_\delta) \quad \left\{= sc\frac{d_q}{d_p} = sc/\gamma\right\}$
$\hat{r} = y_{p1}\delta(1+\epsilon+\epsilon_\delta)$	$\hat{r} = y_{q1}\delta(1+\epsilon+\epsilon_\delta)$
$\hat{d}_p^{2\,\prime} = \frac{d_p^2}{\delta}(1+\epsilon+\epsilon_\delta) \quad \left\{= c^2 d_p^2\right\}$	$\hat{d}_p^{2\,\prime} = \frac{d_q^2}{\delta}(1+\epsilon+\epsilon_\delta) \quad \{= s^2 d_q\}$
$\hat{d}_q^{2\,\prime} = d_p^2\delta(1+\epsilon+\epsilon_\delta) \quad \left\{= \frac{d_p^2}{c^2}\right\}$	$\hat{d}_q^{2\,\prime} = d_p^2\delta(1+\epsilon+\epsilon_\delta) \quad \left\{= \frac{d_p^2}{s^2}\right\}$

indicative of unavoidable information loss in the calculation. For sufficiently small angles, $|(\hat{\delta}(\hat{\alpha}\hat{\beta}) - \delta(\alpha\beta))/\delta(\alpha\beta)| \le \mathbf{u}$, which insures high accuracy for $\hat{r}$, $\hat{d}_p^{2\,\prime}$ and $\hat{d}_q^{2\,\prime}$. An analogous formulation holds for the $\frac{\pi}{2} > |\theta| > \frac{\pi}{4}$ case.

In Table 4, the computed constants and associated errors, followed by the merging of the diagonal weight matrix are shown. In the small angle case, the ϵ_c term is introduced to facilitate the trigonometric substitution in the analysis. It also shows that the algebraic representation of a small number subtracted from 1 is indirectly similar to the ϵ_δ term since ϵ_c tends to diminish for progressively smaller angles. Correspondingly, in the large angle case, the ϵ_s term, analogous to the ϵ_c term of the small angle analysis, will behave similar to the ϵ_δ term by tending to diminish corresponding to progressively larger angles.

These equations correspond closely to Parlett's results for the standard slow and standard fast rotations. Extracting the error terms for the small and large angle rotations, respectively, we have the equations,

$$(9) \qquad \begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} = \begin{bmatrix} d_p'y_p' \\ d_q'y_q' \end{bmatrix} + \begin{bmatrix} (2\epsilon+\epsilon_\delta)c & (6\epsilon+\epsilon_\delta)s \\ -(6\epsilon+2\epsilon_\delta)s & (2\epsilon+\epsilon_\delta+\epsilon_c)c \end{bmatrix} \begin{bmatrix} d_p y_p \\ d_q y_q \end{bmatrix},$$

TABLE 4

Self-scaling fast rotation vector floating point intermediate errors

$\lvert\theta\rvert \le \frac{\pi}{4},\ \lvert d_p\rvert \ge \lvert d_q\rvert$	$\frac{\pi}{2} > \lvert\theta\rvert > \frac{\pi}{4},\ \lvert d_p\rvert < \lvert d_q\rvert$
$\begin{aligned} \hat{y}_p{}' &= [y_p + \hat{\beta} y_q(1+\epsilon)](1+\epsilon) \\ &= y_p(1+\epsilon) + \beta y_q(1+5\epsilon) \\ \hat{d}_p{}'\hat{y}_p{}' &= \hat{d}_p{}' y_p(1+\epsilon) \\ &\quad + \hat{d}_p{}' t \tfrac{d_q}{d_p} y_q(1+5\epsilon) \\ &= c d_p y_p(1+2\epsilon+\epsilon_\delta) \\ &\quad + s d_q y_q(1+6\epsilon+\epsilon_\delta) \end{aligned}$	$\begin{aligned} \hat{y}_p{}' &= [y_q + \hat{\beta} y_p(1+\epsilon)](1+\epsilon) \\ &= y_q(1+\epsilon) + \beta y_p(1+5\epsilon) \\ \hat{d}_p{}'\hat{y}_p{}' &= \hat{d}_p{}' y_q(1+\epsilon) + \hat{d}_p{}' t^{-1} \tfrac{d_p}{d_q} y_p(1+5\epsilon) \\ &= s d_q y_q(1+2\epsilon+\epsilon_\delta) \\ &\quad + c d_p y_p(1+6\epsilon+\epsilon_\delta) \end{aligned}$
$\begin{aligned} \hat{y}_q{}' &= [y_q - \hat{\alpha}\hat{y}_p{}'(1+\epsilon)](1+\epsilon) \\ &= y_q(1+\epsilon) - \alpha\beta y_q(1+9\epsilon+\epsilon_\delta) \\ &\quad -\alpha y_p(1+5\epsilon+\epsilon_\delta) \\ &= (1-\alpha\beta(1+8\epsilon+\epsilon_\delta)) y_q(1+\epsilon) \\ &\quad -\alpha y_p(1+5\epsilon+\epsilon_\delta) \\ &\quad \{\alpha\beta = \tau\beta/\delta = c^2\tau\beta = c^2t^2 = s^2\} \\ &= (1-s^2(1+8\epsilon+\epsilon_\delta)) y_q(1+\epsilon) \\ &\quad -\alpha y_p(1+5\epsilon+\epsilon_\delta) \\ &\quad \{\text{let } (1-s^2(1+\epsilon)) = c^2(1+\epsilon_c)\} \\ &= c^2 y_q(1+\epsilon+\epsilon_c) \\ &\quad -cs\tfrac{d_p}{d_q} y_p(1+5\epsilon+\epsilon_\delta) \\ \hat{d}_q{}'\hat{y}_q{}' &= c d_q y_q(1+2\epsilon+\epsilon_\delta+\epsilon_c) \\ &\quad -s d_p y_p((1+6\epsilon+2\epsilon_\delta) \end{aligned}$	$\begin{aligned} \hat{y}_q{}' &= [-y_p + \hat{\alpha}\hat{y}_p{}'(1+\epsilon)](1+\epsilon) \\ &= -y_p(1+\epsilon) + \alpha\beta y_p(1+9\epsilon+\epsilon_\delta) \\ &\quad +\alpha y_q(1+5\epsilon+\epsilon_\delta) \\ &= -(1-\alpha\beta(1+8\epsilon+\epsilon_\delta)) y_p(1+\epsilon) \\ &\quad +\alpha y_q(1+5\epsilon+\epsilon_\delta) \\ &\quad \{\alpha\beta = \tau\beta/\delta = s^2\tau\beta = s^2t^{-2} = c^2\} \\ &= -(1-c^2(1+8\epsilon+\epsilon_\delta)) y_p(1+\epsilon) \\ &\quad +\alpha y_q(1+5\epsilon+\epsilon_\delta) \\ &\quad \{\text{let } (1-c^2(1+\epsilon)) = s^2(1+\epsilon_s)\} \\ &= -s^2 y_p(1+\epsilon+\epsilon_c) \\ &\quad +sc\tfrac{d_q}{d_p} y_q(1+5\epsilon+\epsilon_\delta) \\ \hat{d}_q{}'\hat{y}_q{}' &= -s d_p y_p(1+2\epsilon+\epsilon_\delta+\epsilon_s) \\ &\quad +c d_q y_q((1+6\epsilon+2\epsilon_\delta) \end{aligned}$

$$(10)\qquad \begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} = \begin{bmatrix} d_p' y_p' \\ d_q' y_q' \end{bmatrix} + \begin{bmatrix} (2\epsilon+\epsilon_\delta)s & (6\epsilon+\epsilon_\delta)c \\ (6\epsilon+2\epsilon_\delta)c & -(2\epsilon+\epsilon_\delta+\epsilon_s)s \end{bmatrix} \begin{bmatrix} d_q y_q \\ d_p y_p \end{bmatrix},$$

which may be represented respectively as the following two inequalities,

$$(11)\qquad \left\| \begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} - \begin{bmatrix} d_p' y_p' \\ d_q' y_q' \end{bmatrix} \right\| \le \left\| \begin{bmatrix} (2\epsilon+\epsilon_\delta)c & (6\epsilon+\epsilon_\delta)s \\ -(6\epsilon+2\epsilon_\delta)s & (2\epsilon+\epsilon_\delta+\epsilon_c)c \end{bmatrix} \right\| \cdot \left\| \begin{bmatrix} d_p y_p \\ d_q y_q \end{bmatrix} \right\|,$$

$$(12)\qquad \left\| \begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} - \begin{bmatrix} d_p' y_p' \\ d_q' y_q' \end{bmatrix} \right\| \le \left\| \begin{bmatrix} (2\epsilon+\epsilon_\delta)s & (6\epsilon+\epsilon_\delta)c \\ (6\epsilon+2\epsilon_\delta)c & -(2\epsilon+\epsilon_\delta+\epsilon_s)s \end{bmatrix} \right\| \cdot \left\| \begin{bmatrix} d_q y_q \\ d_p y_p \end{bmatrix} \right\|.$$

An analogous analysis of the two alternate small and large angle formulae, respectively, yields the similar inequalities:

$$(13)\quad \left\|\begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} - \begin{bmatrix} d'_p y'_p \\ d'_q y'_q \end{bmatrix}\right\| \leq \left\|\begin{bmatrix} (2\epsilon+\epsilon_\delta+\epsilon_c)c & (8\epsilon+2\epsilon_\delta)s \\ -(4\epsilon+\epsilon_\delta)s & (2\epsilon+\epsilon_\delta)c \end{bmatrix}\right\| \cdot \left\|\begin{bmatrix} d_p y_p \\ d_q y_q \end{bmatrix}\right\|,$$

$$(14)\quad \left\|\begin{bmatrix} \hat{d}_p{}'\hat{y}_p{}' \\ \hat{d}_q{}'\hat{y}_q{}' \end{bmatrix} - \begin{bmatrix} d'_p y'_p \\ d'_q y'_q \end{bmatrix}\right\| \leq \left\|\begin{bmatrix} (2\epsilon+\epsilon_\delta+\epsilon_s)s & (8\epsilon+2\epsilon_\delta)c \\ (4\epsilon+\epsilon_\delta)c & -(2\epsilon+\epsilon_\delta)s \end{bmatrix}\right\| \cdot \left\|\begin{bmatrix} d_q y_q \\ d_p y_p \end{bmatrix}\right\|,$$

Remarkably, the r calculation involved no floating point operations, only a direct copy of y_{pi} or y_{qi} for the small and large angle calculations respectively. The above analysis shows that the large angle formulation symmetrically reflects the identical behavior of the small angle formulation. In addition to the bounding of the change of magnitude of the weights at each rotation, the correct selection of rotation respective of angle can minimize ϵ_c, ϵ_s, and ϵ_δ. For extreme angles, e.g. angles generated by rows having one weight much greater than the other, the 2ϵ terms will dominate the error, yielding high accuracy. The above analysis shows that the ordering of weightings is inconsequential. If the scalars are computed with extended precision, the error bound will be tighter at a negligible computational work overhead for larger matrices, as most of the work of the algorithm is in the vector computations.

The following example illustrates the effects of the row sorting in Givens rotations with extreme weights.

Let

$$A(\eta) = \begin{bmatrix} a_{pp} & a_{pq} \\ \eta a_{qp} & \eta a_{qq} \end{bmatrix} \text{ and } \bar{A}(\eta) = \begin{bmatrix} \eta\bar{a}_{pp} & \eta\bar{a}_{pq} \\ \bar{a}_{qp} & \bar{a}_{qq} \end{bmatrix}.$$

where $|a_{ij}| \approx O(1)$, for all i, j. The Givens transformations that insure $a'_{qp} = 0$ and $\bar{a}'_{qp} = 0$ in $A' = G(\eta)A(\eta)$ and $\bar{A}' = \bar{G}(\eta)\bar{A}(\eta)$, respectively, are

$$\begin{array}{rclcrcl} G(\eta) & = & \frac{1}{r}\begin{bmatrix} a_{pp} & \eta a_{qp} \\ -\eta a_{qp} & a_{pp} \end{bmatrix} & & \bar{G}(\eta) & = & \frac{1}{\bar{r}}\begin{bmatrix} \eta\bar{a}_{pp} & \bar{a}_{qp} \\ -\bar{a}_{qp} & \eta\bar{a}_{pp} \end{bmatrix} \\ r = a'_{pp} & = & \sqrt{a_{pp}^2+\eta^2 a_{qp}^2} & \text{and} & \bar{r} = \bar{a}_{pp} & = & \sqrt{\eta^2\bar{a}_{pp}^2+\bar{a}_{qp}^2} \\ a'_{pq} & = & (a_{pp}a_{pq}+\eta^2 a_{qq}a_{qp})/r & & \bar{a}'_{pq} & = & (\eta^2\bar{a}_{pp}\bar{a}_{pq}+\bar{a}_{qq}\bar{a}_{qp})/\bar{r} \\ a'_{qq} & = & \eta(a_{pp}a_{qq}-a_{qp}a_{pq})/r. & & \bar{a}'_{qq} & = & \eta(\bar{a}_{pp}\bar{a}_{qq}-\bar{a}_{qp}\bar{a}_{pq})/\bar{r}. \end{array}$$

For sufficiently large $\eta \gg 1$,

$$\begin{array}{rclcrcl} |a'_{pq}| & \approx & |\eta a_{qq}| & \text{and} & |\bar{a}'_{pq}| & \approx & |\eta\bar{a}_{pq}| \\ |a'_{qq}| & \approx & |a_{pp}a_{qq}/a_{qp} - a_{pq}|, & & |\bar{a}'_{qq}| & \approx & |\bar{a}_{qq} - \bar{a}_{qp}\bar{a}_{pq}/\bar{a}_{pp}|. \end{array}$$

For sufficiently small $0 < \eta \ll 1$,

$$\begin{array}{rcl} |a'_{pq}| & \approx & |a_{pq}| \\ |a'_{qq}| & \approx & |\eta(a_{qq} - a_{qp}a_{pq}/a_{pp})|, \end{array} \quad \text{and} \quad \begin{array}{rcl} |\bar{a}'_{pq}| & \approx & |\bar{a}_{qq}| \\ |\bar{a}'_{qq}| & \approx & |\eta(\bar{a}_{pp}\bar{a}_{qq}/\bar{a}_{qp} - \bar{a}_{pq})|. \end{array}$$

In each of the above four cases, information from the row with larger value dominates the resultant superior row.

Also, note that in each case, the more heavily weighted row of the resultant matrix is in the superior position regardless of its initial location. The implication of this property is that a sequence of rotations will move the greater row towards the top of the matrix.

This analysis is corroborated by our experimental evidence. Additionally, error analysis for the self-scaling fast rotation shows the importance of applying the appropriate large or small angle rotation and rotation parameter calculation, as the use of the incorrect algorithm would violate the bounds and amplify floating point errors.

4 Column Pivoting and Multiple Constraints

In the previous section, we have shown that fast rotations are not sensitive to row sorting. This is not the case when Householder transformation is used for the QR decomposition. A 4×3 exemplary matrix in [PR69] demonstrates the poor accuracy which resulted from an improper row sorting when a Householder QR decomposition is performed. In [VL85], there are examples that illustrates that B-over-A as shown in (3) produces much more accurate solutions than A-over-B when Householder QR decomposition is performed.

Another example in [VL85] shows that the column pivoting is also necessary for better accuracy in the solution. The following theorem due to Stewart [Ste84] illuminates the importance of the column pivoting in weighting method.

THEOREM 1. *Let the weighted augmented matrix be*

$$\begin{bmatrix} B \\ A \end{bmatrix} = \begin{bmatrix} B_1 & B_2 \\ \epsilon A_1 & \epsilon A_2 \end{bmatrix}, \; B_1 \in \mathbf{R}^{l \times l}, \tag{15}$$

and let R_{11} be the triangular factor of the QR decomposition of B_1.

Then the QR decomposition of $\begin{bmatrix} B \\ A \end{bmatrix}$ *is*

$$\begin{bmatrix} B_1 & B_2 \\ \epsilon A_1 & \epsilon A_2 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} + \mathcal{O}(\epsilon^2) & R_{11}^{-1} B_2 + \mathcal{O}(\epsilon^2) \\ 0 & \epsilon \bar{R}_{22} + \mathcal{O}(\epsilon^3) \end{bmatrix}, \tag{16}$$

where $\bar{A}_2 = \bar{Q}_{22}\bar{R}_{22}$ *is the QR decomposition of* $\bar{A}_2 = A_2 - A_1 B_1^{-1} B_2$*, with*

$$Q_1 = \begin{bmatrix} B_1 R_{11}^{-1} + \mathcal{O}(\epsilon^2) \\ \epsilon A_1 R_{11}^{-1} + \mathcal{O}(\epsilon^3) \end{bmatrix} \quad \text{and } Q_2 = \begin{bmatrix} -\epsilon B_1^{-T} A_1^T + \mathcal{O}(\epsilon^3) \\ I + \mathcal{O}(\epsilon^2) \end{bmatrix} \bar{Q}_{22}. \tag{17}$$

□

The keystone of this theorem is that the submatrix B_1 must be well conditioned for a stable algorithm, which corroborates the proof of Powell and Reid in [PR69] that column pivoting should be used in solving the WLS problem. This also shows that the solution by the weighting method gives an approximation to the solution by the direct elimination method. Therefore, column pivoting is also important in the weighting method. In fact, when $\eta \to \infty$, in the weighting method, the direct elimination method is obtained. Note that the Givens rotations are applied to weighted and unweighted row pairs in

$$\frac{1}{\sqrt{\eta^2 a_{pi}^2 + a_{qi}^2}} \begin{bmatrix} \eta a_{pi} & a_{qi} \\ -a_{qi} & \eta a_{pi} \end{bmatrix} \begin{bmatrix} \eta a_{pi} \\ a_{qi} \end{bmatrix} = \begin{bmatrix} \eta a'_{pi} \\ 0 \end{bmatrix} \tag{18}$$

becomes

$$\begin{bmatrix} 1 & 0 \\ -a_{qi}/a_{pi} & 1 \end{bmatrix} \begin{bmatrix} a_{pi} \\ a_{qi} \end{bmatrix} = \begin{bmatrix} a'_{pi} \\ 0 \end{bmatrix} \tag{19}$$

after a row and column scaling as $\eta \to \infty$, which is a Gaussian elimination step in the direct elimination method.

Applying column pivoting to the weighting method essentially does not change the computational complexity. However, in many applications, such as adaptive beamforming, the LSE needs to be solved for many different constraint matrices B for each fixed data matrix A. As mentioned in [VL85], the A matrix can be first orthogonally triangularized as

$$A = QR.$$

The computational cost of this operation will be, for $m \geq n$, $\frac{1}{2}\nu n^2\left(m-\frac{n}{3}\right)$ flops. [2] Then for each constraint matrix B, we can triangularize the matrix

$$C = \begin{pmatrix} \eta B \\ R \end{pmatrix} \tag{20}$$

For $m \gg n$, the precomputation of the QR decomposition of A will result in significant savings for the direct elimination and weighting methods. However, the orthogonal transformations from the right in the null space method may lead to a complete fill in of the empty half of the triangular matrix. Thus, the floating point complexity of the null space method for each new set of equality constraints will be $\mathcal{O}\left(n^3\right)$, which is significantly more work than what is required by the other two methods.

We can triangularize (20) by first applying the orthogonal transformations to B matrix to make it upper trapezoidal, in $\frac{1}{2}\nu l^2\left(n-\frac{l}{3}\right)$ flops. Then the direct elimination and the weighting methods have $\mathcal{O}\left(ln\right)$ elements to annihilate at a expense of $\frac{1}{2}\nu l\left(n\left(n-l\right)+\frac{l^2}{3}\right)$ flops. The direct elimination method is a little less expensive because $\frac{l^2}{2}$ elements can be eliminated via elementary transformations.

We now discuss how the column pivoting can be efficiently incorporated in the weighting method for solving the multiple constraint problem. We first obtain the QR decomposition of the A matrix. In the next process of triangularizing the matrix C, column pivoting may significantly increase the computational complexity since it can destroy the triangular structure of R. However, the constraint matrix B typically has a very small number of rows as in beamforming. Since we want to have well conditioned matrix B_1, we can apply column pivoting only in the first l steps of triangularizing the matrix C. The orthogonal transformations necessary to triangularize C with this pivoting will introduce at most ln extra nonzero elements to annihilate. Thus, we can achieve better stability without increasing the complexity significantly.

5 Numerical Results

In our numerical implementations, we used the fast rotations in two ways–initialized the diagonal factor matrix D as the square of the weights and also $D = I$ with the

[2] ν represents the number of flops used to rotate a single pair of elements ($\nu = 6$–slow, $\nu = 4$–fast)

weights premultiplied into the matrix. Keeping the squared weights has the advantage that it eliminates the square root operation in the fast rotations.

We compare the self-scaling fast rotations with the standard fast plane rotation and with the standard plane rotation, as well as with the Householder's QR method and with the MGS QR method. We used two different elimination orderings combined with merged and non-merged weights for both the standard and self scaling fast rotations, yielding eight fast Givens algorithms. Our computations were performed in Matlab v4 on a Sun Sparc which utilizes IEEE floating-point arithmetic. We tested our algorithms on the three LSE test problems in [VL85] and on the Powell and Reid matrix [PR69]. Following [VL85], we used a large weight $\eta \gg 1$ with B rather than a small weight $\epsilon \ll 1$ with A, and contrasted the accuracy of computing with A *over* B versus B *over* A to compare the effects of row sorting in different algorithms.

The reference value of x_{LSE} is computed via a generalized singular value decomposition (GSVD) of A and B

$$U^T AX = D_A = \mathrm{diag}(\alpha_1, \ldots, \alpha_n), V^T BX = D_B = \mathrm{diag}(\beta_1, \ldots, \beta_l),$$

where $U \equiv [u_1, \ldots, u_m] \in \mathbb{R}^{m \times m}, V \equiv [v_1, \ldots, v_l] \in \mathbb{R}^{l \times l}$ are orthogonal and $X \equiv [x_1, \ldots, x_n] \in \mathbb{R}^{n \times n}$ is nonsingular, $0 = \alpha_1 = \ldots = \alpha_q < \alpha_{q+1} \leq \alpha_{l+1} \leq \ldots \leq \alpha_n$, and $\beta_1 \geq \ldots \beta_l > 0$

When

$$y_{\mathrm{LSE}} = \left(\frac{v_1^T d}{\beta_1}, \ldots, \frac{v_l^T d}{\beta_l}, \frac{u_{l+1}^T b}{\alpha_{l+1}}, \ldots, \frac{u_n^T b}{\alpha_n} \right)^T, \tag{21}$$

$x_{\mathrm{LSE}} = X y_{\mathrm{LSE}}$ [VL85]. The solution, $x(\eta)$, to the WLS (2), is

$$x(\eta) = x_{\mathrm{LSE}} + \sum_{i=q+1}^{l} \frac{\gamma_i^2}{\eta^2 + \gamma_i^2} \cdot \frac{\rho_i}{\alpha_i} x_i, \tag{22}$$

where $\gamma_i \equiv \frac{\alpha_i}{\beta_i}$, $\rho_i = u_i b - \gamma_i v_i^T d$, and $q = \dim(\mathcal{N}(A)) = \dim(\mathrm{span}(x_1, \ldots, x_q))$.

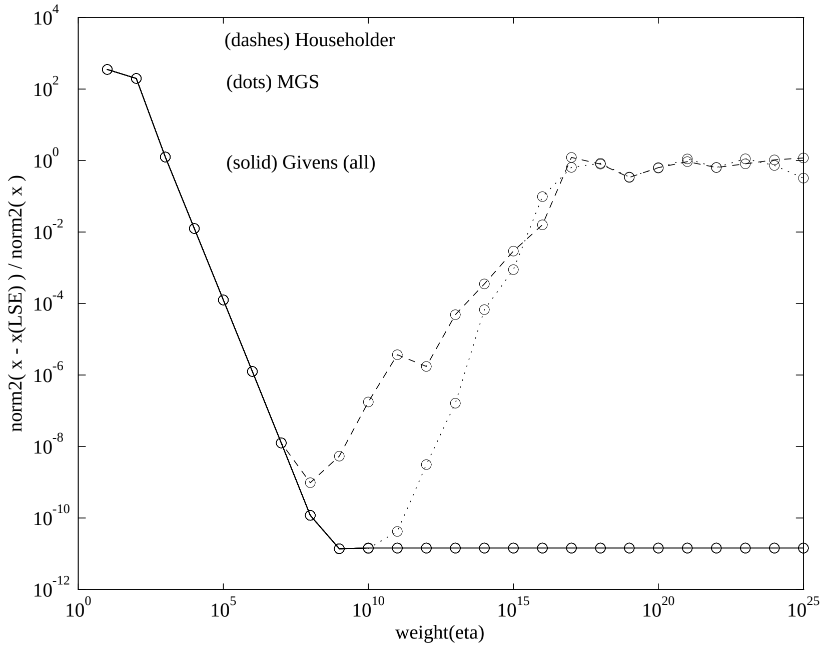
The results we achieved for Householder QR decomposition are commensurate with those listed in [VL85] for all three problems. The error, $\|\bar{x}(\eta) - x(\eta)\|_2$, where $\bar{x}(\eta)$ is the computed approximation to $x(\eta)$ (Figure 1b), shows the Householder's QR method significantly losing accuracy starting at $\eta = 10^7$ for the third test problem ($l = 2, m = 6, n = 4$).

In [BP92], the MGS method is shown to be numerically equivalent to the Householder method applied to a matrix with the $n \times n$ zero matrix adjoined to the top. In our tests, the MGS method also begins significantly losing accuracy starting at $\eta = 10^{12}$. However, the Givens QR decompositions maintain accuracy for all of the tested η's. All of the QR methods give good results for satisfying the constraint equations at all weights (Figure 1c). However, both the MGS and Householder methods begin decaying in satisfying the $Ax = b$ equation at $\eta = 10^{16}$ (figure 1d), whereas the Givens methods remain accurate. The implication of these data is that to the limit of our tests on these three test matrices, only the Givens QR methods may be employed without the possibility of overshooting the optimum weight when using WLS to solve LSE problems. These tests did not illuminate any significant difference in the accuracies between the different Givens methods and rotation orderings. Similar tests on a sequence of stiff randomly generated matrices with varying structures corroborated these results. The tests examined the results of each of 12 orthogonalization algorithms on 6 random matrices of dimension $\{m = 24, n = 6, l = 4\}$ having 9 different row orderings for each of a wide range of weightings. Additionally, tests on the Powell and Reid matrix in Matlab showed exceptionally good accuracy by the Givens methods for all possible row permutations.

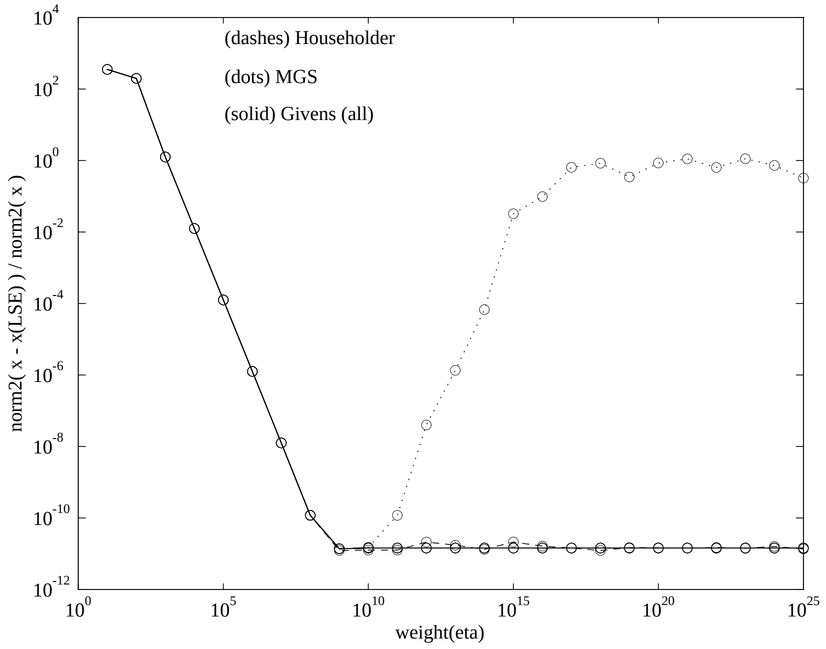
6 Summary

We described the self-scaling fast Givens rotation for solving the LSE problem by the method of extreme weighting. We subsequently presented an error analysis of the self-scaling square root free fast rotation which shows its stability and its row ordering invariance. We also showed row ordering invariance with extremely disparate weights. The complications involved in column pivoting, which is necessary for increased stability, is examined. We presented the results of our numerical experiments which showed that for a large spread of row weights, self-scaling fast Givens rotations exhibit superior accuracy to Householder and modified Gram-Schmidt decompositions for arbitrary row orderings. The numerical results showed that once a high accuracy was achieved, the self-scaling rotations maintained that accuracy for much larger weights, thus obviating the need for iterating with insufficient row weights as described in [VL85].

Acknowledgement: We thank the referee for valuable suggestions, and Prof. Barlow for valuable discussions regarding Givens rotations for extreme row weights.

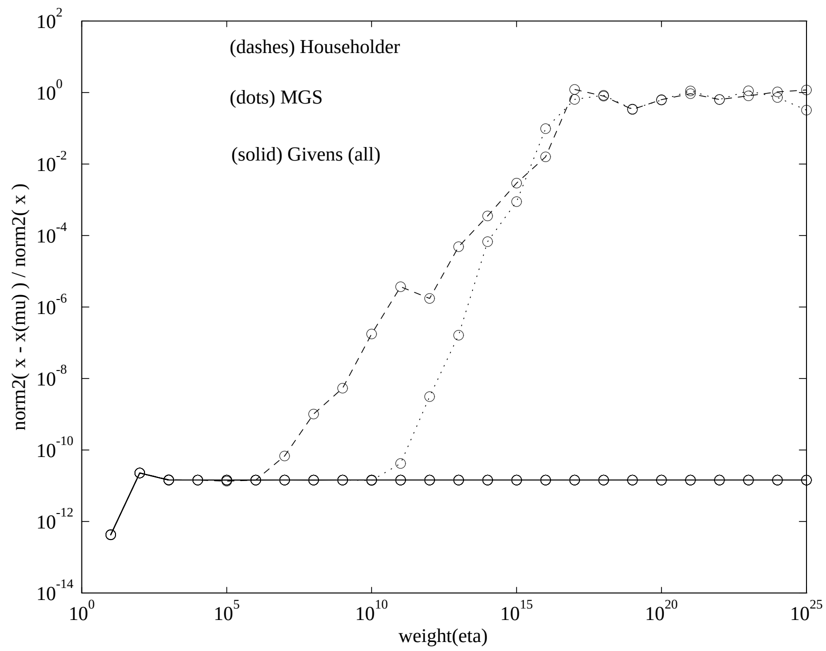


(a) $\min_x \left\| \begin{bmatrix} A \\ \eta B \end{bmatrix} x - \begin{bmatrix} b \\ \eta d \end{bmatrix} \right\|_2, \eta \gg 1$

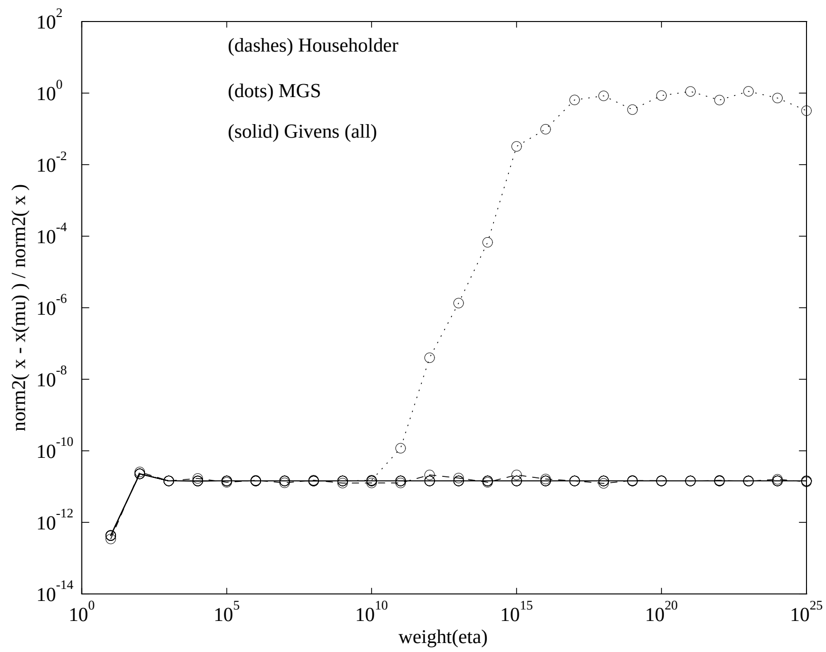


(b) $\min_x \left\| \begin{bmatrix} \eta B \\ A \end{bmatrix} x - \begin{bmatrix} \eta d \\ b \end{bmatrix} \right\|_2, \eta \gg 1$

FIG. 1. $\|x - x(LSE)\|_2 \,/\, \|x\|_2$: *[VL85]'s third matrix using column pivoting*

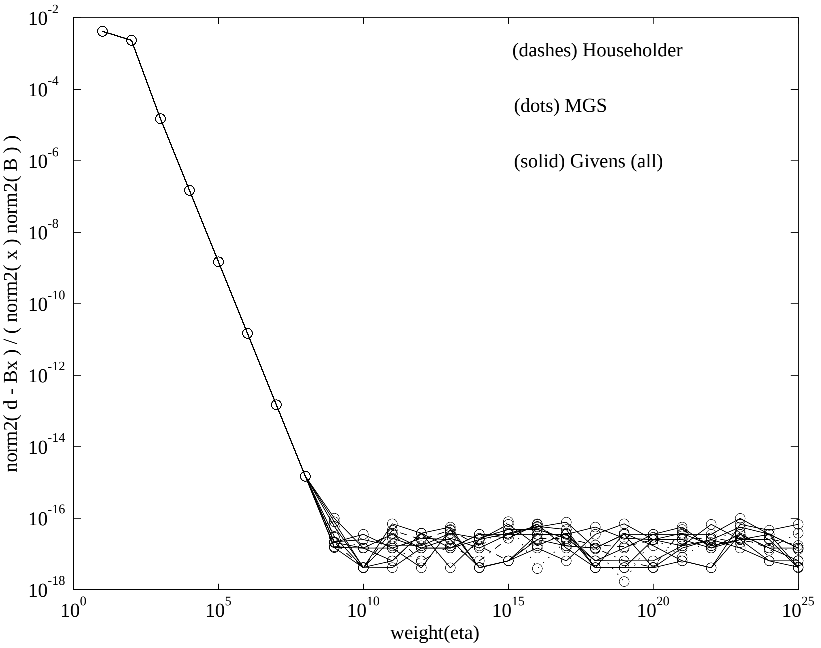


(a) $\min_x \left\| \begin{bmatrix} A \\ \eta B \end{bmatrix} x - \begin{bmatrix} b \\ \eta d \end{bmatrix} \right\|_2, \eta \gg 1$

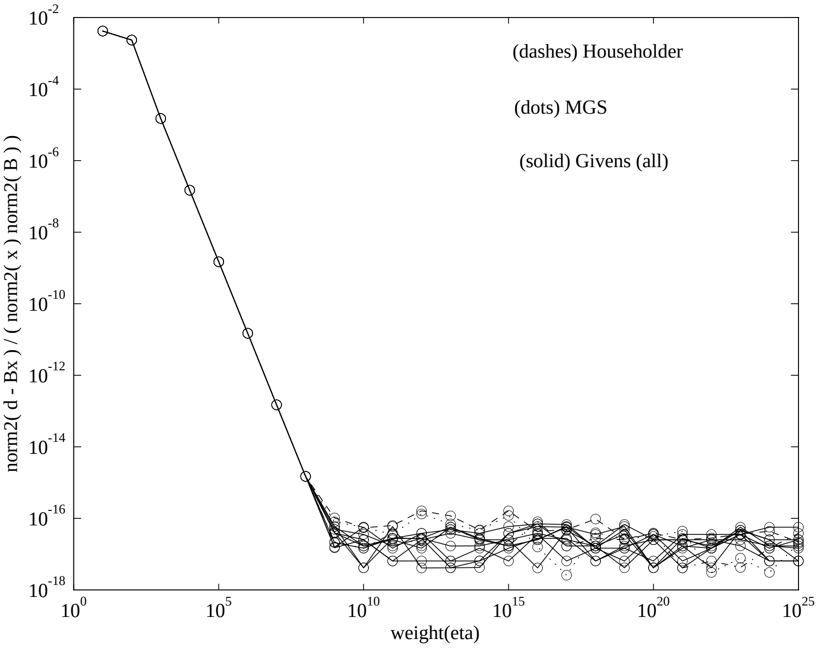


(b) $\min_x \left\| \begin{bmatrix} \eta B \\ A \end{bmatrix} x - \begin{bmatrix} \eta d \\ b \end{bmatrix} \right\|_2, \eta \gg 1$

FIG. 2. $\|x - x(\mu)\|_2 \,/\, \|x\|_2$: *[VL85]'s third matrix, with column pivoting*

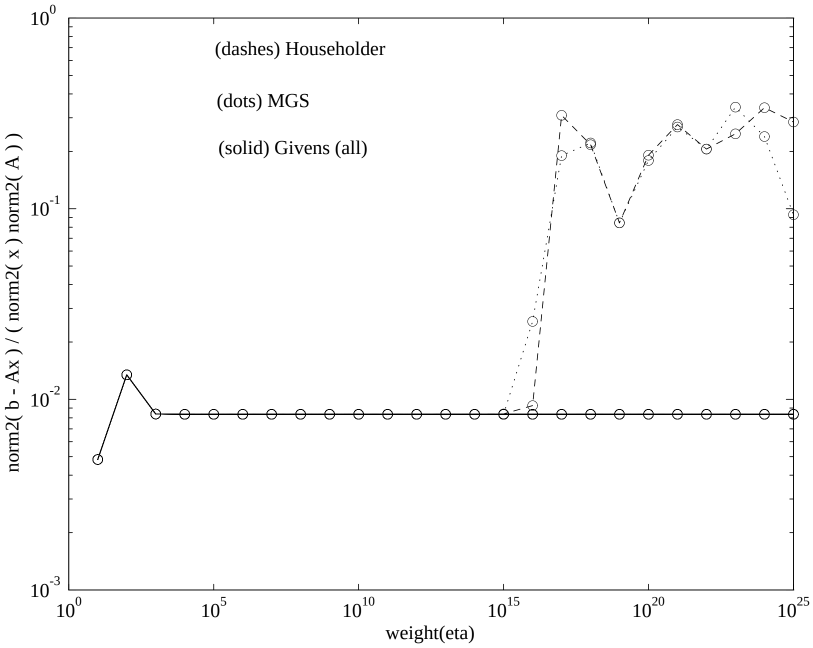


(a) $\min_x \left\| \begin{bmatrix} A \\ \eta B \end{bmatrix} x - \begin{bmatrix} b \\ \eta d \end{bmatrix} \right\|_2, \eta \gg 1$

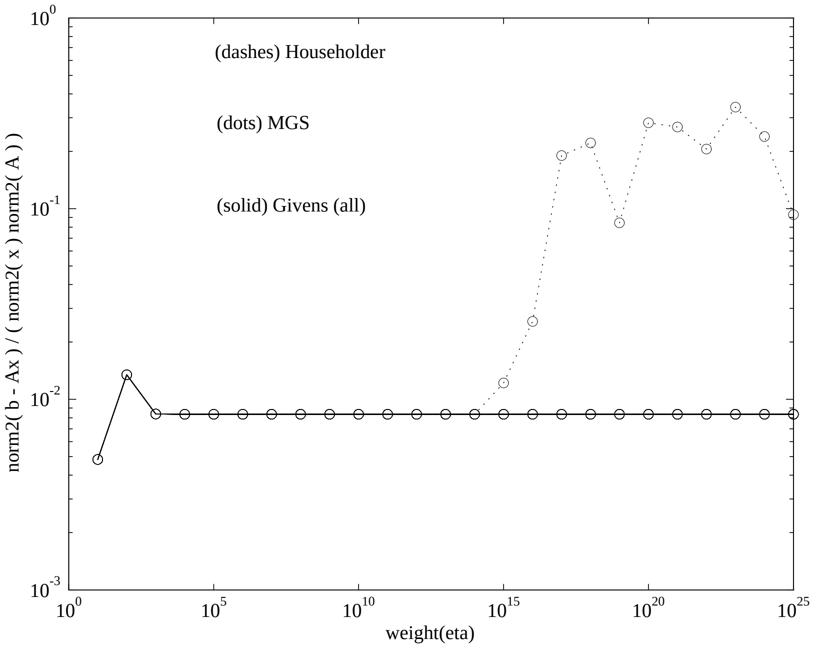


(b) $\min_x \left\| \begin{bmatrix} \eta B \\ A \end{bmatrix} x - \begin{bmatrix} \eta d \\ b \end{bmatrix} \right\|_2, \eta \gg 1$

FIG. 3. $\|x - x(d - Bx)\|_2 / (\|x\|_2 \|B\|_2)$: *[VL85]'s third matrix, with column pivoting*



(a) $\min_x \left\| \begin{bmatrix} A \\ \eta B \end{bmatrix} x - \begin{bmatrix} b \\ \eta d \end{bmatrix} \right\|_2, \eta \gg 1$



(b) $\min_x \left\| \begin{bmatrix} \eta B \\ A \end{bmatrix} x - \begin{bmatrix} \eta d \\ b \end{bmatrix} \right\|_2, \eta \gg 1$

FIG. 4. $\|x - x(b - Ax)\|_2 / (\|x\|_2 \|A\|_2)$: *[VL85]'s third matrix, with column pivoting*

References

[AP] A.A. Anda and H. Park. Implementation of self-scaling fast rotations on vector architectures. In preparation.

[AP94] A.A. Anda and H. Park. Fast plane rotation algorithms with dynamic scaling. *SIAM J. Matrix Anal. Appl.*, 15(1):162–174, Jan 1994.

[Bar88] J. Barlow. Error analysis and implementation aspects of deferred correction for equality constrained least squares problems. *SIAM J. Numer. Anal.*, 25(6):1340–1358, Dec 1988.

[BI87] J. Barlow and I. Ipsen. Scaled Givens rotations for the solution of linear least squares problems on systolic arrays. *SIAM J. Sci. Stat. Comput.*, 8(5):716–733, sep 1987.

[Bjö84] Å. Björck. A general updating algorithm for constrained linear least squares problems. *SIAM J. Sci. Stat. Comput.*, 5(2):394–402, jun 1984.

[Bjö90] Å. Björck. Least squares methods. In P. G. Ciarlet and J. L. Lions, editors, *Handbook of Numerical Analysis, V.1: Solution of Equations in* R^n, pages 465–652. Elsevier Science Publishers B.V. (North-Holland), 1990.

[Bjö91] Å. Björck. Algorithms for linear least squares problems. Technical report, Department of Mathematics, Linköping University, August 1991.

[BNP88] J.L. Barlow, N.K. Nichols, and R.J. Plemmons. Iterative methods for equality constrained least squares problems. *SIAM J. Sci. Stat. Comput.*, 9(5):892–906, Sep 1988.

[BP92] Å. Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm. *SIAM J. Matrix Anal. Appl.*, 13:176–190, 1992.

[dR89] P. de Rijk. A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer. *SIAM J. Sci. Stat. Comput.*, 10(2):359–371, mar 1989.

[Eld80] L. Eldén. Perturbation theory for the least squares problem with linear equality constraints. *SIAM J. Numer. Anal.*, 17(3):338–350, 1980.

[Eld82] L. Eldén. A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT*, 22:487–502, 1982.

[Gen73] W.M. Gentleman. Least squares computations by Givens transformations without square roots. *J. Inst. Maths Applics*, 13:329–336, 1973.

[GVL89] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences. Johns Hopkins Press, Baltimore, MD, second edition, 1989.

[GW92] M. Gulliksson and P.-Å. Wedin. Modifying the QR-decomposition to constrained and weighted linear least squares. *SIAM J. Matrix Anal. Appl.*, 13(4):1298–1313, Oct 1992.

[Ham74] S. Hammarling. A note on modifications to the Givens plane rotation. *J. Inst. Maths Applics*, 13:215–218, 1974.

[LH74] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems.* Prentice Hall Series in Automatic Computation. Prentice Hall, Englewood Cliffs, NJ, 1974.

[LHKK79a] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh. Algorithm 539: Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Soft.*, 5:324–325, 1979.

[LHKK79b] C.L. Lawson, R.J. Hanson, D.R. Kincaid, and F.T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Soft.*, 5:308–323, 1979.

[LW70] Ö. Leringe and P.-Å. Wedin. A comparison between different methods to compute a vector x which minimizes $\|Ax - b\|$, when $Gx = h$. Technical report, Department of Computer Science, Lund University, 1970.

[Par80] B.N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall Series in Computational Mathematics. Prentice Hall, Englewood Cliffs, N.J., first edition, 1980.

[PR69] M. Powell and J. Reid. On applying Householder's method to linear least squares problems. In A. Morell, editor, *Proceedings IFIP Congress 1968*, pages 122–126, Amsterdam, 1969. North-Holland.

[Rat82] W. Rath. Fast Givens rotations for orthogonal similarity transformations. *Numer. Math.*, 40:47–56, 1982.

[Ste76] G. W. Stewart. The economical storage of plane rotations. *Numer. Math.*, 25:137–138, 1976.

[Ste84] G. W. Stewart. On the asymptotic behavior of scaled singular value and QR decompostions. *Math. Comp.*, 43(168):483–489, Oct 1984.

[Vav92] S. A. Vavasis. Stable numerical algorithms for equilibrium systems. Technical Report 92-1280, Department of Computer Science, Cornell University, Ithaca, NY, 1992. Accepted to *SIAM J. Matrix Analysis.*

[VL83] C.F. Van Loan. A generalized SVD analysis of some weighting methods for equality constrained least squares. In Pite Havsbad, B. Kågström, and A. Ruhe, editors, *Matrix Pencil Proceedings, Lecture Notes in Mathematics 973*, pages 245–262, New York, 1983. Springer-Verlag.

[VL85] C.F. Van Loan. On the method of weighting for equality constrained least squares problems. *SIAM J. Numer. Anal.*, 22:851–864, 1985.