SOLUTION OF AUGMENTED LINEAR SYSTEMS USING ORTHOGONAL FACTORIZATIONS *

Å. BJÖRCK† AND C. C. PAIGE‡

Abstract. We consider solving $x + Ay = b$ and $A^T x = c$ for given b, c and $m \times n$ A of rank n. This is called the augmented system formulation (ASF) of two standard optimization problems, which include as special cases the minimum 2-norm solution of a linear underdetermined system ($b = 0$) and the linear least squares problem ($c = 0$), as well as more general problems. We examine the numerical stability of methods (for the ASF) based on the QR factorization of A, whether by Householder transformations, Givens rotations, or the modified Gram-Schmidt (MGS) algorithm, and consider methods which use Q and R, or only R. We discuss the meaning of stability of algorithms for the ASF in terms of stability of algorithms for the underlying optimization problems.

We prove the backward stability of several methods for the ASF which use Q and R, including a new one based on MGS [A. Björck and C. C. Paige, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 176–190], and also show under what circumstances they may be regarded as strongly stable. We show why previous methods using Q from MGS were not backward stable, but illustrate that some of these methods may be "acceptable-error stable" (a strengthening of the weak stability criterion in [J. R. Bunch, Linear Algebra Appl., 88/89 (1987), pp. 49–66]). We point out that the numerical accuracy of methods that do not use Q does not depend to any significant extent on which of the above three QR factorizations is used. We then show that the standard methods which do not use Q are not backward stable or even acceptable-error stable for the general ASF problem, and discuss how iterative refinement can be used to counteract these deficiencies.

Key Words. orthogonalization, QR factorization, least squares, minimum norm solution, semi-normal equations, numerical stability, Gram-Schmidt, augmented systems.

AMS Subject Classifications: 65F25, 65G05, 65F05, 65F20

1. Introduction. We consider the problem of solving the symmetric indefinite linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \tag{1.1}$$

involving $A \in \mathbf{R}^{m \times n}$ with rank $n \leq m$. When $c = 0$ this has been called the augmented system formulation of the linear least squares problem, and in general we can call it the Augmented System Formulation (ASF) of the following two problems, since it represents the necessary and sufficient conditions for their solution:

$$\min_x \|b - x\|_2 \quad \text{subject to} \quad A^T x = c, \tag{1.2}$$

$$\min_y \{\|b - Ay\|_2^2 + 2c^T y\}. \tag{1.3}$$

The ASF can be obtained by differentiating the Lagrangian $\|b - x\|_2^2 + 2y^T(A^T x - c)$ of (1.2), and equating to zero. Here y is the vector of Lagrange multipliers. The ASF can also be obtained by differentiating (1.3) to give $A^T(b - Ay) = c$, and setting x to be the residual $x = b - Ay$. We will refer

* This research was partially supported by NSERC of Canada Grant No. A9236.

† Department of Mathematics, Linköping University, S-581 83, Linköping, Sweden, akbjo@math.liu.se).

‡ Department of Computer Science, McGill University, Montreal, Quebec, Canada, H3A 2A7, (chris@cs.mcgill.ca).

to (1.2) when we are only interested in x, (1.3) when we are only interested in y, and (1.1) when we want to consider x and y.

Setting $b = 0$ in (1.2), and so in (1.1), gives the problem of finding the minimum 2-norm solution of a Linear Underdetermined System (LUS), which occurs for example as a subproblem in optimization algorithms (see for example [10]). Setting $c = 0$ in (1.3) gives the much used Linear Least Squares (LLS) problem. Thus the ASF covers two important special cases. It occurs in its full form (1.1) in the iterative refinement of least squares solutions [5], and has been studied extensively in its form (1.2) in [3, 2]. A more general form of (1.1),

$$\begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \tag{1.4}$$

with H symmetric and positive semidefinite, occurs in many application areas, since it represents the condition for equilibrium of a physical system, see Strang [16]. The system (1.4) is also called a saddle point system.

The ASF can be rephrased as the *normal equations*

$$A^TAy = A^Tb - c, \qquad x = b - Ay. \tag{1.5}$$

If c lies in $\mathcal{R}(A^T)$, the range of A^T, for *general* $A \in \mathbf{R}^{m\times n}$, then the singular value decomposition $A = U_1\Sigma_1V_1^T$ with square Σ_1 gives

$$V_1\Sigma_1^2V_1^Ty = V_1\Sigma_1U_1^Tb - c,$$

so for any $y_{\mathcal{N}(A)} \in \mathcal{N}(A)$, the null space of A, a solution is $y = \hat{y} + y_{\mathcal{N}(A)}$ with

$$\begin{aligned} \hat{y} = V_1\Sigma_1^{-1}U_1^Tb - V_1\Sigma_1^{-2}V_1^Tc &= A^\dagger b - (A^TA)^\dagger c, \\ x = (I - U_1U_1^T)b + U_1\Sigma_1^{-1}V_1^Tc &= P_{\mathcal{N}(A^T)}b + (A^\dagger)^Tc. \end{aligned} \tag{1.6}$$

The case of rank A less than n is discussed in [3], but here we restrict ourselves to $A \in \mathbf{R}^{m\times n}$ with rank n, so that (1.1) is always consistent with a unique solution.

For problems with a sparse matrix A, one effective way of solving (1.1) is by computing a sparse LDL^T decomposition of the augmented matrix using 1×1 and 2×2 pivots, see [1]. Here we concentrate on methods which use some QR factorization

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1R, \tag{1.7}$$

where Q is orthogonal and R upper triangular. The Modified Gram-Schmidt algorithm (MGS), and QR factorizations via Givens rotations or Householder transformations, in theory all compute this factorization. In practice if the condition number $\kappa = \kappa(A) \equiv \sigma_1/\sigma_n$ is large, ($\sigma_1 \geq \ldots \geq \sigma_n$ being the singular values of A), then the columns of Q_1 produced by MGS are not particularly orthogonal [4]. If orthogonality is crucial, then in the past the other two algorithms have been used.

In [7] we showed some important properties of MGS in the presence of rounding errors, and also indicated how Q_1 loses orthogonality in MGS. We showed that the R obtained from MGS is numerically about as good as that obtained using rotations or Householder transformations, and that the loss of orthogonality in Q_1 from MGS can effectively be compensated for in computations which use Q_1, without altering the MGS algorithm or reorthogonalizing the columns of Q_1. As a practical example of this, we showed how Q_1 and R from MGS may be used to solve the augmented system (1.1) reliably, despite the loss of orthogonality in Q_1. Here we give the rounding error analysis for this, examine this augmented system more closely, and discuss some other relevant algorithms in detail.

Using Q_1 and R in the factorization (1.7), it is easy to see that in theory one algorithm for solving (1.1) is

$$z = R^{-T}c, \qquad d = Q_1^T b, \qquad x = b - Q_1(d - z), \qquad y = R^{-1}(d - z). \tag{1.8}$$

When Q, and not only Q_1, is available, we can use the very similar approach

$$z = R^{-T}c, \qquad \begin{pmatrix} d \\ f \end{pmatrix} = Q^T b, \qquad x = Q\begin{pmatrix} z \\ f \end{pmatrix}, \qquad y = R^{-1}(d - z). \tag{1.9}$$

The norms and functionals in (1.2) and (1.3) are then available from

$$\|b - x\|_2 = \|d - z\|_2, \quad \|b - Ay\|_2^2 = \|f\|_2^2 + \|z\|_2^2, \quad c^T y = z^T(d - z).$$

Note that either x or y can be computed without the other, so we can solve (1.2), (1.3), or both to give (1.1).

In Section 2 we discuss the different forms of stability we encounter in this paper, and also provide the perturbation analysis results we will need to prove acceptable-error stability for some algorithms for the ASF. It suffices for this overview to realize that *backward stable* algorithms give small residuals, while *acceptable-error stable* algorithms give *error* norms which are comparable with those given by backward stable algorithms, but can give significantly worse *residuals*. We point out how our definition of acceptable-error stability differs from weak stability in [8].

In Section 3 we show that (1.9) using Givens rotations or Householder transformations is backward stable for the ASF, and we examine under what conditions it also gives the exact answer for a nearby problem having exactly the same structure as the ASF (1.1), that is, when it is *strongly* stable (see [8]) for the ASF. In Section 4 we summarize the important result that MGS is *numerically* equivalent to the Householder QR factorization applied to A augmented with a square block of zeros on top, see [7]. In [7] we produced a new backward stable algorithm for (1.1) using Q_1 and R from MGS, and showed how it specializes to LUS and LLS and solution of square nonsingular systems. In Section 5 we prove this algorithm is backward stable for the ASF, giving rounding error bounds which are nearly as good as those using Householder transformations, and consider under what conditions this algorithm is also strongly stable for the ASF. In Section 6 we indicate how the old use of MGS tended to lose backward stability, but that algorithms using this old approach in solving the ASF could still be acceptable-error stable.

Using R from the QR factorization, (1.5) can be rewritten as the *seminormal equations*

$$R^T R y = A^T b - c, \qquad x = b - Ay. \tag{1.10}$$

If A has large dimensions, then it is often uneconomical to store and access the matrix Q, or even Q_1, and one may choose to use the above seminormal equations to compute y and x. Then all three QR factorization techniques give similar accuracy, and we illustrate some limitations of this accuracy for this class of problems, and how these can be overcome to some extent. In Section 7 we extend earlier results for the seminormal equations method (1.10) discussed in [14] for LUS, and [6] for LLS, to the full problem (1.1). We show the seminormal equations method is neither backward nor acceptable-error stable for the general ASF problem. However iterative refinement can be expected to produce acceptable-error stable answers with only one step of iterative refinement when $\kappa^2 u \leq 1$, u being the unit roundoff, or machine precision. We hope to provide a full error analysis showing this at a later date.

2. Some forms of numerical stability. The algorithms we will deal with exhibit different forms of numerical stability, so we will use the following notions. A *backward stable* algorithm is one which is guaranteed to give the exact answer to a nearby problem. This is the concept developed and used so effectively by Wilkinson (see, e.g., [17]), and for which the term *stable* is used in [8]. The other two basic ideas we use are those of *strong stability* and *acceptable-error stability.* Below we use *block strongly stable* to mean a certain limited form of strong stability for the ASF.

2.1. Strong stability. Following [8] we will say an algorithm for solving *linear equations* is *strongly stable* for a class of matrices $\mathcal{M}$ if for each $M \in \mathcal{M}$, and for each g, the computed solution $\bar{z}$ to $Mz = g$ satisfies $\bar{M}\bar{z} = \bar{g}$, where $\bar{M} \in \mathcal{M}$, and $\bar{M}$ and $\bar{g}$ are close to M and g respectively. In addition we will say an algorithm is (backward or strongly) stable *in the extended sense* when it is guaranteed to give an answer which is *very close* to the exact answer of the relevant nearby system, see Stewart [15, p. 76].

For the ASF we will choose the class of matrices $\mathcal{M}$ to have both the zero and unit block structure *and* the symmetry of ASF matrices, so that a strongly stable algorithm for solving (1.1) would give the exact answer to a problem

$$\begin{pmatrix} I & A+\delta A_2 \\ (A+\delta A_1)^T & 0 \end{pmatrix} \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} b+\delta b \\ c+\delta c \end{pmatrix}, \qquad , \tag{2.1}$$

with satisfactory bounds on the sizes of δA_1, δA_2, δb, δc, and

$$\delta A \equiv \delta A_1 = \delta A_2. \tag{2.2}$$

We will assume each $A + \delta A_i$ has rank n. The term *block strongly stable* will correspond to (2.1) only.

Strong stability is a very important concept for the ASF as we now show. Consider problems close to (1.2) and (1.3)

$$\min_x \|(b+\delta b) - x\|_2, \qquad (A+\delta A_1)^T x = c + \delta c, \tag{2.3}$$

$$\min_y \{\|(b+\delta b) - (A+\delta A_2)y\|_2^2 + 2(c+\delta c)^T y\}. \tag{2.4}$$

with satisfactory bounds on δA_1, δA_2, δb, and δc. If an algorithm for (1.1) always computes $\bar{x}$ and $\bar{y}$ satisfying (2.1) with (2.2), so that it is strongly stable for (1.1), then $\bar{x}$ is the solution and $\bar{y}$ the vector of Lagrange multipliers for (2.3) with $\delta A_1 \equiv \delta A$, but also $\bar{y}$ is the solution and $\bar{x}$ the residual for (2.4) with $\delta A_2 \equiv \delta A$. That is the algorithm is backward stable for both the solution *and* Lagrange multipliers in (1.2), and the solution *and* residual in (1.3). The reverse clearly also holds, giving the important equivalence:

LEMMA 2.1. *The following three statements about a given algorithm are equivalent in that if one is true they are all true:*

- *The algorithm is strongly stable for computing* $\begin{pmatrix} x \\ y \end{pmatrix}$ *in (1.1).*
- *The algorithm is backward stable for computing the solution x and vector of Lagrange multipliers y in (1.2).*
- *The algorithm is backward stable for computing the solution y and residual $x = b - Ay$ in (1.3).*

But an algorithm for (1.2) may compute only x, and an algorithm for (1.3) may compute only y. We see a backward stable algorithm for finding only x in (1.2) computes $\bar{x}$ solving (2.3). Differentiating the Lagrangian of this shows there must also *exist* $\bar{y}$ so that (2.1) holds with (2.2). Although the algorithm does not find the full solution of (1.1), it can be thought of as an algorithm for computing

x in (1.1), and we can then say it is strongly stable for computing only x in (1.1). On the other hand if we can compute $\bar{x}$ satisfying (2.1) with (2.2), this implies there exists a corresponding $\bar{y}$, so $\bar{x}$ solves (2.3) and the algorithm is backward stable for (1.2). This and a similar argument for (1.3) give the following:

LEMMA 2.2. *An algorithm for computing only x in (1.2) is backward stable for (1.2) if and only if it is strongly stable for computing only x in (1.1).*

LEMMA 2.3. *An algorithm for computing only y in (1.3) is backward stable for (1.3) if and only if it is strongly stable for computing only y in (1.1).*

It is clear from this discussion that if our main aim is to solve the optimization problem (1.2) or (1.3), then a strongly stable algorithm for (1.1) is highly desirable. However this is not always easy to obtain, and we will discuss to what extent some algorithms approach this. It appears that most backward stable algorithms for the ASF which take account of the blocks of (1.1) compute $\bar{x}$ and $\bar{y}$ satisfying (2.1) but not (2.2). These give strong stability for the class of matrices with the zero and unit block structure of the ASF, but without the symmetry. Since this is a stronger concept of stability than ordinary backward stability, for brevity we will refer to such algorithms as *block strongly stable* for the ASF. We remark that in [9] it was proven that any algorithm which is backward stable for the class of symmetric positive definite matrices or the class of nonsingular symmetric matrices will also be strongly stable on the same matrix class. It does not seem possible to prove a similar property for algorithms for the matrix class ASF in general because of the additional zero and unit matrix subblock structure.

When Q and R are computed by Householder transformations, an error analysis of (1.9) was given in [5, see (5.23)] to show it is block strongly stable in the extended sense for computing both x and y in (1.1). Later an analysis was given in [3] to show the stronger result that it is backward stable in the extended sense for computing only x in (1.2), and so from Lemma 2.2 we now see it is also strongly stable for computing only x in (1.1). In fact if the QR factorization is computed by Givens rotations or Householder transformations, then we will use standard results in Section 3 to produce a new simple analysis showing that the method (1.9) is not only backward stable for (1.2) and block strongly stable for (1.1), but can be made *strongly* stable for (1.1) (and so backward stable for (1.3)) in that the computed $\bar{x}$ and $\bar{y}$ are exact for (2.1), with bounds

$$\|\delta A_i\|_2 \leq c_A u \|A\|_2, \qquad \|\delta c\|_2 \leq c_c u \|c\|_2, \qquad \|\delta b\|_2 \leq c_b u \|b\|_2, \tag{2.5}$$

where u is the machine precision. Here c_A, c_b, and c_c are modest constants depending on the dimensions m, n and the details of the arithmetic. The exact sizes of these and similar constants are not very important, and can be found from the analysis and the literature. Using these bounds leads to three sets of perturbation results for the three related problems (1.1)–(1.3).

2.2. Acceptable-error stability. We also define an *acceptable-error stable* algorithm to be one which gives a solution whose error size is never significantly worse than the error bound obtained from a tight perturbation analysis of a backward stable algorithm for the same problem. That is, the norm of the error (but not necessarily the residual) in an acceptable-error stable algorithm will not be significantly worse in magnitude than that of the worst error that could result from a backward stable algorithm applied to a problem having the same condition and magnitude of data. Note this is different from the notion of weak stability in [8], where the error bound need only be comparable in size to that of a backward stable algorithm when the problem is well conditioned [8, p.63, Def. 3]. With that definition, an algorithm which only ensured (2.10), see ahead, held with κ replaced by κ^2 would apparently be weakly stable, as would the normal equations method for LLS. It follows that the main notions of stability we have mentioned can be ordered in increasing desirability as *weak, acceptable-error, backward,* and *strong*. For some problems we may also include some limited form of strong stability between backward and strong stability, for example our "block" strongly stable for the ASF.

The different forms of backward stability raise the question as to which form should be used for error comparison in defining acceptable-error stability. Since we will only be examining algorithms which use the block structure of the ASF, we will use block strongly stable algorithms for comparison. We will see this leads to the same results as using strongly stable algorithms for comparison, but it is straightforward to see requiring no structure at all in the error of the ASF matrix would lead to weaker results.

First we bound the sizes of the residuals in (2.1) for a block strongly stable method, note from the top block we have $b - \bar{x} - A\bar{y} = \delta A_2 \bar{y} - \delta b$, and with (2.5)

$$\|b - \bar{x} - A\bar{y}\|_2 = \|\delta b - \delta A_2 \bar{y}\|_2 \leq u(c_A \|A\|_2 \|\bar{y}\|_2 + c_b \|b\|_2). \tag{2.6}$$

Similarly for the bottom block $c + \delta c = (A + \delta A_1)^T \bar{x}$, so $c - A^T \bar{x} = \delta A_1^T \bar{x} - \delta c$, giving

$$\|c - A^T \bar{x}\|_2 = \|\delta A_1^T \bar{x} - \delta c\|_2 \leq u(c_A \|A\|_2 \|\bar{x}\|_2 + c_c \|c\|_2). \tag{2.7}$$

A perturbation analysis for x in (1.2) was given in [3], however we also need the analysis to include y in the ASF. Defining $\bar{x} = x + \delta x$, $\bar{y} = y + \delta y$, we can move the δA_i terms to the right hand side of (2.1) and subtract (1.1) to give

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \begin{pmatrix} \delta b - \delta A_2 \bar{y} \\ \delta c - \delta A_1^T \bar{x} \end{pmatrix},$$

which from (1.6) gives

$$\begin{aligned} \delta y &= A^\dagger(\delta b - \delta A_2 \bar{y}) + (A^T A)^{-1}(\delta A_1^T \bar{x} - \delta c), \\ \delta x &= P_{\mathcal{N}(A^T)}(\delta b - \delta A_2 \bar{y}) - (A^\dagger)^T(\delta A_1^T \bar{x} - \delta c). \end{aligned}$$

Note that the two terms that are summed to give δx are orthogonal, just as for x in (1.6). Ignoring the orthogonality, we have

$$\|\delta x\|_2 \leq u\beta, \quad \|\delta y\|_2 \leq \sigma_n^{-1} u\beta, \quad u\beta \equiv \|\delta b - \delta A_2 \bar{y}\|_2 + \sigma_n^{-1} \|\delta A_1^T \bar{x} - \delta c\|_2. \tag{2.8}$$

This with (2.6) and (2.7) gives the perturbation bounds for (2.1)

$$\begin{aligned} \|\bar{x} - x\|_2 \leq u\beta, & \qquad \|\bar{y} - y\|_2 \leq \sigma_n^{-1} u\beta, \\ \beta \leq c_b \|b\|_2 + c_A \|A\|_2 \|\bar{y}\|_2 & \quad + \quad \sigma_n^{-1} c_c \|c\|_2 + \kappa c_A \|\bar{x}\|_2, \end{aligned} \tag{2.9}$$

which is the same as if we had assumed (2.2), and shows how an acceptable-error stable algorithm for the ASF should behave.

If we eliminate $\bar{y}$ using $(A + \delta A_2)\bar{y} = (b + \delta b) - \bar{x}$ in considering the solution $\bar{x}$ of (1.2) only, we have when $c_A u\kappa < 1$ that $(A + \delta A_2)$ has rank n so $\bar{y}$ is unique, and

$$\|\bar{y}\|_2 \leq \sigma_n^{-1}(1 - c_A u\kappa)^{-1}(\|b - \bar{x}\|_2 + c_b u \|b\|_2).$$

Using this in (2.9) shows that if (1.2) is perturbed according to (2.5), and $2c_A u\kappa \leq 1$,

$$\|\bar{x} - x\|_2 \leq u\Big(2c_b \|b\|_2 + 2\kappa c_A \|b - \bar{x}\|_2 + \sigma_n^{-1} c_c \|c\|_2 + \kappa c_A \|\bar{x}\|_2\Big). \tag{2.10}$$

The bound in (2.10) is similar to the bound in [3]. It follows that an acceptable-error stable algorithm for x in (1.2), and so in LUS, would be one where the computed solution $\bar{x}$ was guaranteed to have an error $\|\bar{x} - x\|_2$ with a bound having a similar form to this.

On the other hand, if we eliminate $\bar{x} = (b + \delta b) - (A + \delta A_2)\bar{y}$ in (2.9) in considering the solution of (1.3) only, we obtain $\|\bar{x}\|_2 \le \|b - A\bar{y}\|_2 + c_b u\|b\|_2 + c_A u\|A\|_2\|\bar{y}\|_2$, so if $\kappa c_A u \le 1$

$$\|\bar{y} - y\|_2 \le \sigma_n^{-1} u\Big(2c_b\|b\|_2 + 2c_A\|A\|_2\|\bar{y}\|_2 + \sigma_n^{-1}c_c\|c\|_2 + \kappa c_A\|b - A\bar{y}\|_2\Big). \tag{2.11}$$

Again this shows how an acceptable-error stable algorithm for (1.3), and so for LLS, must behave. If (1.3) is perturbed according to (2.5) the bound (2.11) shows that a large "residual" $\|b - A\bar{y}\|_2$ brings in a κ/σ_n effect, just as in LLS.

The result (2.9) shows that in ill-conditioned problems with reasonable $\|A\|_2$, we expect $\|\bar{y} - y\|_2$ to be significantly larger than $\|\bar{x} - x\|_2$, although the relative errors could be similar, see (1.6). The bounds in (2.10) and (2.11) differ from (2.9) in that only the computed $\bar{x}$ is used in (2.10), and only the computed $\bar{y}$ is used in (2.11). The bounds in (2.9) show how an acceptable-error stable algorithm for the ASF must behave, but since they are tighter bounds than (2.10) and (2.11), an algorithm that is acceptable-error stable for the ASF is also acceptable-error stable for (1.2) and (1.3), and so for LUS and LLS.

It will facilitate some later comparisons if we simplify the bound in (2.7). To do this note from $c = (A + \delta A_1)^T\bar{x} - \delta c$ that $\|c\|_2 \le (1 + c_A u)\|A\|_2\|\bar{x}\|_2 + c_c u\|c\|_2$, giving

$$\|c\|_2 \le \frac{1 + c_A u}{1 - c_c u}\|A\|_2\|\bar{x}\|_2, \tag{2.12}$$

and the simplified bound

$$\|c - A^T\bar{x}\|_2 = \|\delta A_1^T\bar{x} - \delta c\|_2 \le u c_{c,A}\|A\|_2\|\bar{x}\|_2, \quad c_{c,A} \equiv \frac{c_A + c_c}{1 - c_c u}. \tag{2.13}$$

The weaker versions of (2.10) and (2.11) are then, if $\kappa c_{c,A} \le 1$,

$$\|\bar{x} - x\|_2 \le u\Big(2c_b\|b\|_2 + 2\kappa c_A\|b - \bar{x}\|_2 + \kappa c_{c,A}\|\bar{x}\|_2\Big), \tag{2.14}$$

$$\|\bar{y} - y\|_2 \le \sigma_n^{-1} u\Big(2c_b\|b\|_2 + 2c_A\|A\|_2\|\bar{y}\|_2 + \kappa c_{c,A}\|b - A\bar{y}\|_2\Big). \tag{2.15}$$

3. Stability of a method for the ASF. Björck [5] showed (1.9) is block strongly stable in the extended sense for (1.1) using the Householder QR factorization. An analysis is given in [3] to show the same algorithm is backward stable in the extended sense for computing only x in (1.2), and [3] also contains a perturbation analysis to bound the error in the computed $\bar{x}$ in (1.2) for such backward stable algorithms. Here we give a simple new analysis and consider the possibility of *strong* stability for the algorithm (1.9) for the ASF (1.1). As we saw in Section 2.1 strong stability ensures backward stability for both x and y in (1.2) and (1.3). This analysis will also be used in Section 5 to prove the stability of an MGS method.

From [17] we can quickly derive each of the following results using the model of rounding in [11, §2.4.2], where $\bar{Q}, \bar{R}, \bar{z}, \bar{d}, \bar{f}, \bar{x}, \bar{y}$ are the computed results of (1.9) using rotations or Householder transformations in (1.7).

$$A + E_1 = \tilde{Q}\begin{pmatrix}\bar{R}\\ 0\end{pmatrix}, \qquad \tilde{Q} = (\tilde{Q}_1, \tilde{Q}_2) = \bar{Q} + E_2, \tag{3.1}$$

$$(\bar{R} + E_3)^T\bar{z} = c, \qquad (\bar{R} + E_6)\bar{y} = \bar{d} - \bar{z}, \tag{3.2}$$

$$\begin{pmatrix}\bar{d}\\ \bar{f}\end{pmatrix} = (\tilde{Q} + E_4)^T b, \qquad \bar{x} = (\tilde{Q} + E_5)\begin{pmatrix}\bar{z}\\ \bar{f}\end{pmatrix}, \tag{3.3}$$

where $\tilde{Q}$ is exactly orthogonal, and

$$\|E_i\|_2 \le c_i u\|A\|_2, \quad i = 1, 3, 6; \qquad \|E_j\|_2 \le c_j u, \quad j = 2, 4, 5.$$

Note we have written the main equations in terms of the unknown $\tilde{Q} = (\tilde{Q}_1, \tilde{Q}_2)$ rather than the computed $\bar{Q}$. This simplifies the analysis. Putting these results together gives

$$\begin{pmatrix} I_n & 0 & \bar{R}+E_6 \\ 0 & I_{m-n} & 0 \\ (\bar{R}+E_3)^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{z} \\ \bar{f} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} (\tilde{Q}+E_4)^T b \\ c \end{pmatrix}. \tag{3.4}$$

To simplify the analysis further, from the expression for $\bar{x}$ we can define $\tilde{x}$ which is very close to $\bar{x}$ as follows

$$\tilde{x} \equiv \tilde{Q} \begin{pmatrix} \bar{z} \\ \bar{f} \end{pmatrix} = \tilde{Q}(\tilde{Q}+E_5)^{-1}\bar{x} = (I + E_5\tilde{Q}^T)^{-1}\bar{x},$$

$$\|\tilde{x} - \bar{x}\|_2 = \|E_5 \tilde{Q}^T \tilde{x}\|_2 \le c_5 u \left\| \begin{pmatrix} \bar{z} \\ \bar{f} \end{pmatrix} \right\|_2 = c_5 u \|\tilde{x}\|_2 \le \tfrac{c_5 u}{1-c_5 u}\|\bar{x}\|_2.$$

Now multiplying by $\tilde{Q}$ in the appropriate places in (3.4), and using (3.1), gives

$$\begin{pmatrix} I & (A+E_1+\tilde{Q}_1 E_6) \\ (A+E_1+\tilde{Q}_1 E_3)^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} (I+\tilde{Q}E_4^T)b \\ c \end{pmatrix}. \tag{3.5}$$

which is similar to (2.1). This shows the algorithm is block strongly stable in the extended sense, but by replacing $\tilde{x}$ by $(I+E_5\tilde{Q}^T)^{-1}\bar{x}$ it is straightforward to show that it is also block strongly stable in the ordinary sense. This has a slightly different form to the result in [5, (5.23)& Thm. 1]. One advantage of this simple and elegant form is that it shows just how close the algorithm is to being strongly stable in the extended sense. The error matrices E_3 and E_6 arise from solution of the triangular systems in (3.2). Solutions of triangular systems are almost invariably computed to high accuracy, [17, p. 251], and E_3 and E_6 will tend to be small. This can be proved for several special classes of triangular matrices, see [13]. If they were zero (or equal) then the algorithm would be strongly stable in the extended sense. By using iterative refinement with double precision accumulation of inner-products as in [17] we could compute $\bar{z}$ and $\bar{y}$ which would be exact to the limit of precision for the stored $\bar{R}$, c and $\bar{d} - \bar{z}$, and the algorithm would effectively be strongly stable in the extended sense, implying backward stability for both x and y in (1.2) and (1.3), see Lemma 2.1.

Another advantage of such a clean analysis is it makes the following obvious. If we only need x in (1.2), then the second solve in (3.2) need not be carried out, but we can theoretically *define* a $\tilde{y}$ satisfying $(\bar{R}+E_3)\tilde{y} = \bar{d} - \bar{z}$. Using this instead of the second equation in (3.2) gives (3.4) and (3.5) with E_6 and $\bar{y}$ replaced by E_3 and $\tilde{y}$ respectively. This shows the algorithm is strongly stable (in the extended sense) for computing only x in (1.1), and therefore backward stable (in the extended sense) for computing only x in (1.2). We have thus rederived the important result of [3], but in a simple way that we can now use in Section 5.

4. Modified Gram-Schmidt as a Householder method. A rounding error analysis of MGS was given in [4]. There it was shown that assuming $\bar{c}_2\kappa u < 1$, the *computed* factors $\bar{Q}_1$ and $\bar{R}$ satisfy

$$A + \bar{E} = \bar{Q}_1\bar{R}, \qquad \|\bar{E}\|_2 \le \bar{c}_1 u \|A\|_2,$$

$$\|I - \bar{Q}_1^T\bar{Q}_1\|_2 \le \bar{c}_2\kappa u, \tag{4.1}$$

where $\bar{c}_i$ are constants depending on m, n and the details of the arithmetic. Hence $\bar{Q}_1\bar{R}$ accurately represents A and the departure from orthogonality can be bounded in terms of the condition number $\kappa = \sigma_1/\sigma_n$.

It was further shown in [7] how the modified Gram-Schmidt algorithm for the QR factorization can be interpreted as Householder's method applied to the matrix A *augmented with a square matrix* *of zero elements on top.* This is not only true in theory, but in the presence of rounding errors as well. It is sufficient for us here to look at the theoretical result only.

Let $A \in \mathbf{R}^{m\times n}$ have rank n, and let $O_n \in \mathbf{R}^{n\times n}$ be a zero matrix. Consider the two QR factorizations (here we use Q for $m \times m$ and P for $(m+n)\times(m+n)$ orthogonal matrices),

$$A = Q\begin{pmatrix} R \\ 0 \end{pmatrix} = (\,Q_1 \quad Q_2\,)\begin{pmatrix} R \\ 0 \end{pmatrix},$$

$$\tilde{A} \equiv \begin{pmatrix} O_n \\ A \end{pmatrix} = P\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}\begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}. \tag{4.2}$$

Clearly P_{11} is zero, so P_{21} is an $m \times n$ matrix of orthonormal columns, and $A = Q_1 R = P_{21}\tilde{R}$. If upper triangular R and $\tilde{R}$ are both chosen to have positive diagonal elements in $A^TA = R^TR = \tilde{R}^T\tilde{R}$, then $R = \tilde{R}$ by uniqueness, so $P_{21} = Q_1$ can be found from any QR factorization of the augmented matrix. The last m columns of P are then arbitrary to the extent of an $m \times m$ orthogonal multiplier.

The important result is that the *Householder* QR factorization of the augmented matrix is *numerically* equivalent to MGS applied to A, see [7].

P in (4.2) has the form $P = P_1P_2\ldots P_n$, where it was shown in [7] that every Householder transformation P_k is effectively defined by $Q_1 = (q_1, \ldots, q_n)$, since

$$P_k = I - v_kv_k^T, \qquad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}. \tag{4.3}$$

It was shown in [7] that in theory

$$P = \begin{pmatrix} O_n & Q_1^T \\ Q_1 & I - Q_1Q_1^T \end{pmatrix}, \tag{4.4}$$

and the effect of rounding errors on this was given. In particular, it can be shown (this is a simple extension of [7]) that when $\bar{R}$ is computed using MGS, there exists exactly *orthonormal* $\tilde{Q}_1$ such that, for a modest constant c_0,

$$A + E^{(0)} = \tilde{Q}_1\bar{R}, \qquad \tilde{Q}_1^T\tilde{Q}_1 = I, \qquad \|e_j^{(0)}\|_2 \le c_0u\|a_j\|_2, \quad j = 1, \ldots, n, \tag{4.5}$$

and this should be compared with (4.1). In particular this means MGS computes as accurate an $\bar{R}$ as does the QR factorization using Givens rotations or Householder transformations.

A key point is that although the columns $\bar{q}_k$ of the computed $\bar{Q}_1$ in MGS need not be particularly orthonormal, so it is usually inadvisable to use $\bar{Q}_1$ as our set of orthonormal vectors, we *can* use $\bar{P}$ (as the theoretical product of the computed $\bar{P}_k = I - \bar{v}_k\bar{v}_k^T$, $\bar{v}_k^T = (-e_k^T, \bar{q}_k^T)$) to make use of the desired orthogonality. Thus we can solve problems as accurately using MGS as using rotations or Householder transformations for the QR factorization if, instead of using the computed $\bar{Q}_1$ directly, we formulate the problems in terms of (4.2), and use the $\bar{q}_k$ to define $\bar{P}$. Of course in most cases no block of $\bar{P}$ need actually be formed. We use this idea in the next section.

5. Backward stable solution of the ASF using MGS. In Section 3 we saw (1.9) with Givens rotations or Householder transformations on A was block strongly stable for the ASF. However, the same is not true when we use (1.8) with MGS, and we will prove this in the next section. The reason follows from the analysis in Section 3, where we required the computed $\bar{Q}$ to be close to an orthogonal matrix. This orthogonality is not necessarily present in $\bar{Q}_1$ from MGS. Despite this lack of orthogonality, Björck [4] showed the old method using $\bar{Q}_1$ from MGS in LLS is acceptable-error stable. In the next section we will show why the same may be true for the more general problem

(1.1), but since we now know a good algorithm for the ASF based on MGS, see [7], we will give this first, then prove it is block strongly stable, and show when it can be considered strongly stable. These results will also simplify our discussion of stability for the old method using MGS.

Our original ASF (1.1) is equivalent to the augmented system

$$\begin{pmatrix} I_n & 0 & 0 \\ 0 & I_m & A \\ 0 & A^T & 0 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \\ c \end{pmatrix}, \tag{5.1}$$

so applying Householder transformations as in (4.2) gives in theory

$$\begin{pmatrix} I_n & 0 & R \\ 0 & I_m & 0 \\ R^T & 0 & 0 \end{pmatrix} \begin{pmatrix} z \\ v \\ y \end{pmatrix} = \begin{pmatrix} P^T \begin{pmatrix} 0 \\ b \end{pmatrix} \\ c \end{pmatrix} \equiv \begin{pmatrix} d \\ h \\ c \end{pmatrix}, \quad \begin{pmatrix} z \\ v \end{pmatrix} \equiv P^T \begin{pmatrix} w \\ x \end{pmatrix},$$

where from (4.4) z and d are identical to those in (1.8) and (1.9), and $h = Q_2 f$. Clearly $v = h$ in the equation above, and the augmented version of (1.9) is

$$z = R^{-T}c, \quad \begin{pmatrix} d \\ h \end{pmatrix} = P^T \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad \begin{pmatrix} w \\ x \end{pmatrix} = P \begin{pmatrix} z \\ h \end{pmatrix}, \quad y = R^{-1}(d - z). \tag{5.2}$$

Now if we replace A in (3.1) by $\tilde{A}$ in (4.2) etc., the analysis in Section 3 applies to this algorithm here, and so all those stability results also hold for solving (5.1) using Householder transformations. But we saw in Section 4 that MGS is *numerically* equivalent to Householder transformations applied to $\tilde{A}$, and so if we use the q_k from MGS to produce $P_k = I - v_k v_k^T$, $v_k^T = (-e_k^T, q_k^T)$, and use $P^T = P_n \cdots P_2 P_1$ in (5.2), the same stability results hold for solving (5.1) based on MGS. If the triangular solves are carried out sufficiently accurately, then as we showed in Section 3, the methods are effectively strongly stable for (5.1) in the extended sense.

Here is a summary of Algorithm 6.1 in [7] which takes advantage of the structure of the P_k in this numerically stable use of MGS for the ASF. (For a square root free version see Algorithm 6.2 in [7].)

ALGORITHM 5.1.
Backward Stable Algorithm for the ASF based on MGS

1. Carry out MGS on A to give $Q_1 = (q_1, \ldots, q_n)$ and R,
2. Solve $R^T z = c$ for $z = (\zeta_1, \ldots, \zeta_n)^T$,
3. **for** $k = 1, \ldots, n$ **do**$\{\delta_k := q_k^T b; \quad b := b - q_k \delta_k\}$; to give $h := b$, and $d = (\delta_1, \ldots, \delta_n)^T$.
4. **for** $k = n, \ldots, 1$ **do**$\{\omega_k := q_k^T h; \quad h := h - q_k(\omega_k - \zeta_k)\}$; to give $x := h$,
5. Solve $Ry = d - z$ for y.

Note that $w = (\omega_1, \ldots, \omega_n)^T$ is ideally zero, but can be significant when y is large. This algorithm in [7] is new, and different from the MGS algorithms and the Huang algorithm studied in [2].

Although we have shown this is stable for (5.1), this does not prove it is stable for the original ASF in (1.1). The difficulty is caused by the nonzero w. The results analogous to (3.1)–(3.5) are for Algorithm 5.1, where $\bar{P}$, $\bar{R}$, *etc.* denote the computed results, and $\tilde{P}$ is exactly orthogonal:

$$\begin{pmatrix} E_7 \\ A + E_1 \end{pmatrix} = \tilde{P} \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix}, \qquad \tilde{P} = (\tilde{P}_1, \tilde{P}_2) = \begin{pmatrix} \tilde{P}_{11} & \tilde{P}_{12} \\ \tilde{P}_{21} & \tilde{P}_{22} \end{pmatrix} = \bar{P} + E_2, \tag{5.3}$$

$$(\bar{R} + E_3)^T \bar{z} = c, \qquad (\bar{R} + E_6)\bar{y} = \bar{d} - \bar{z}, \tag{5.4}$$

$$(5.5)\qquad \begin{pmatrix} \bar{d} \\ \bar{h} \end{pmatrix} = (\tilde{P}+E_4)^T \begin{pmatrix} 0 \\ b \end{pmatrix}, \qquad \begin{pmatrix} \bar{w} \\ \bar{x} \end{pmatrix} = (\tilde{P}+E_5)\begin{pmatrix} \bar{z} \\ \bar{h} \end{pmatrix},$$

$$(5.6)\qquad \|E_i\|_2 \le c_i u \|A\|_2, \quad i=1,3,6,7; \qquad \|E_i\|_2 \le c_i u, \quad i=2,4,5.$$

Putting these results together gives

$$(5.7)\qquad \begin{pmatrix} I_n & 0 & \bar{R}+E_6 \\ 0 & I_m & 0 \\ (\bar{R}+E_3)^T & 0 & 0 \end{pmatrix} \begin{pmatrix} \bar{z} \\ \bar{h} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} (\tilde{P}+E_4)^T \begin{pmatrix} 0 \\ b \end{pmatrix} \\ c \end{pmatrix}.$$

To simplify the analysis we again define

$$(5.8)\qquad \begin{pmatrix} \tilde{w} \\ \tilde{x} \end{pmatrix} \equiv \tilde{P}\begin{pmatrix} \bar{z} \\ \bar{h} \end{pmatrix} = (I+E_5\tilde{P}^T)^{-1}\begin{pmatrix} \bar{w} \\ \bar{x} \end{pmatrix}, \qquad \left\| \begin{matrix} \tilde{w}-\bar{w} \\ \tilde{x}-\bar{x} \end{matrix} \right\|_2 \le \left\| \begin{matrix} \bar{w} \\ \bar{x} \end{matrix} \right\|_2 \frac{c_5 u}{(1-c_5 u)}.$$

Multiplying by $\tilde{P}$ where necessary, and using (5.3) gives the analogous result to (3.5)

$$(5.9)\qquad \begin{aligned} \begin{bmatrix} I_n & 0 & E_7+\tilde{P}_{11}E_6 \\ 0 & I_m & A+E_1+\tilde{P}_{21}E_6 \\ (E_7+\tilde{P}_{11}E_3)^T & (A+E_1+\tilde{P}_{21}E_3)^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{w} \\ \tilde{x} \\ \bar{y} \end{bmatrix} &= \begin{bmatrix} \delta b_0 \\ b+\delta b_1 \\ c \end{bmatrix}, \\ \begin{bmatrix} \delta b_0 \\ \delta b_1 \end{bmatrix} &\equiv \tilde{P}E_4^T \begin{bmatrix} 0 \\ b \end{bmatrix}. \end{aligned}$$

Again if we do not need y, we can theoretically define a $\tilde{y}$ satisfying $(\bar{R}+E_3)\tilde{y} = \bar{d}-\bar{z}$. Using this instead gives (5.7) and (5.9) with E_6 and $\bar{y}$ replaced by E_3 and $\tilde{y}$ respectively. This shows the algorithm is strongly stable (in the extended sense) for computing only w and x in (5.1), and so from Lemma 2.2 is backward stable (in the extended sense) for computing only w and x in

$$(5.10)\qquad \min_{w,x} \left\| \begin{pmatrix} 0 \\ b \end{pmatrix} - \begin{pmatrix} w \\ x \end{pmatrix} \right\|_2 \quad \text{subject to} \quad \begin{pmatrix} 0 \\ A \end{pmatrix}^T \begin{pmatrix} w \\ x \end{pmatrix} = c,$$

which is in theory just (1.2).

We can also eliminate $\tilde{w}$ from this to obtain an equation of the form of (2.1). This analysis is less elegant than we would like, but since it emphasizes the similar behavior of the MGS and usual Householder approaches, we include it here. From the top block

$$\tilde{w} = -(E_7+\tilde{P}_{11}E_6)\bar{y} + \delta b_0,$$

but from (5.3) and (5.4)

$$(E_7+\tilde{P}_{11}E_6)\bar{y} = \tilde{P}_{11}(\bar{R}+E_6)\bar{y} = \tilde{P}_{11}(\bar{d}-\bar{z}),$$

giving

$$\tilde{w} = \delta b_0 - \tilde{P}_{11}\bar{d} + \tilde{P}_{11}\bar{z}.$$

Now from (5.9) and (5.5)

$$\begin{aligned} \delta b_0 - \tilde{P}_{11}\bar{d} &= (I \quad 0)[\tilde{P}E_4^T - \tilde{P}\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}(\tilde{P}+E_4)^T]\begin{pmatrix} 0 \\ b \end{pmatrix} \\ &= (I \quad 0)\tilde{P}\left[\begin{pmatrix} E_{11}^{(4)} & E_{12}^{(4)} \\ E_{21}^{(4)} & E_{22}^{(4)} \end{pmatrix} - \begin{pmatrix} \tilde{P}_{11} & 0 \\ \tilde{P}_{21} & 0 \end{pmatrix} - \begin{pmatrix} E_{11}^{(4)} & 0 \\ E_{21}^{(4)} & 0 \end{pmatrix} \right]^T \begin{pmatrix} 0 \\ b \end{pmatrix} \\ &= [\tilde{P}_{12}(E_{22}^{(4)})^T - \tilde{P}_{11}\tilde{P}_{21}^T]b \equiv f_1 \quad \text{say}, \end{aligned}$$

and from (5.8)

$$\bar{z} = \tilde{P}_{11}^T \tilde{w} + \tilde{P}_{21}^T \tilde{x},$$

so that under conditions for nonsingularity stated later

$$\begin{aligned} \tilde{w} &= \tilde{P}_{11}\bar{z} + f_1 = \tilde{P}_{11}\tilde{P}_{11}^T\tilde{w} + \tilde{P}_{11}\tilde{P}_{21}^T\tilde{x} + f_1 \equiv \tilde{E}\tilde{x} + \tilde{f} \quad \text{say}, \\ \tilde{E} &\equiv (I - \tilde{P}_{11}\tilde{P}_{11}^T)^{-1}\tilde{P}_{11}\tilde{P}_{21}^T, \qquad \tilde{f} \equiv (I - \tilde{P}_{11}\tilde{P}_{11}^T)^{-1} f_1. \end{aligned} \tag{5.11}$$

We can now use this expression to eliminate $\tilde{w}$ from (5.9). The bottom block of the main equation in (5.9) involves $(E_7 + \tilde{P}_{11}E_3)^T\tilde{w}$, so if we define

$$E^T \equiv (E_7 + \tilde{P}_{11}E_3)^T\tilde{E}, \qquad \delta c \equiv -(E_7 + \tilde{P}_{11}E_3)^T\tilde{f},$$

then after eliminating $\tilde{w}$ we have

$$\begin{pmatrix} I & A + E_1 + \tilde{P}_{21}E_6 \\ (A + E_1 + \tilde{P}_{21}E_3 + E)^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{x} \\ \bar{y} \end{pmatrix} = \begin{pmatrix} b + \delta b_1 \\ c + \delta c \end{pmatrix}, \tag{5.12}$$

$$\begin{aligned} E^T &= (E_7 + \tilde{P}_{11}E_3)^T (I - \tilde{P}_{11}\tilde{P}_{11}^T)^{-1}\tilde{P}_{11}\tilde{P}_{21}^T, \\ \delta c &= -(E_7 + \tilde{P}_{11}E_3)^T (I - \tilde{P}_{11}\tilde{P}_{11}^T)^{-1}[\tilde{P}_{12}(E_{22}^{(4)})^T - \tilde{P}_{11}\tilde{P}_{21}^T]b, \end{aligned}$$

which is our desired format. But $\|\tilde{P}_{21}\|_2 \le 1$, and with (5.6), (5.9),

$$\begin{aligned} \|E_1 + \tilde{P}_{21}E_i\|_2 &\le u(c_1 + c_i)\|A\|_2, \qquad i = 3, 6, \\ \|\delta b_1\|_2 &\le uc_4\|b\|_2, \end{aligned}$$

so it remains for us to bound E and δc.

For simplicity of exposition we assume the condition of the problem is such that

$$\gamma \equiv \max\{c_1, \dots, c_7\} < (4u\kappa)^{-1}, \tag{5.13}$$

as this ensures $\bar{R}+E_3$, $\bar{R}+E_6$, and $\bar{R}$ are all nonsingular — otherwise A could be considered numerically rank deficient — and allows us to simplify some error bounds. Now from (5.3) $\tilde{P}_{11} = E_7\bar{R}^{-1}$, so

$$\begin{aligned} \|\tilde{P}_{11}\|_2 &\le uc_7\sigma_1[\sigma_n - (c_1 + c_7)u\sigma_1]^{-1} \le 2uc_7\kappa \le 1/2, \\ \|(I - \tilde{P}_{11}\tilde{P}_{11}^T)^{-1}\|_2 &\le 4/3, \\ \|(E_7 + \tilde{P}_{11}E_3)^T\|_2 &\le (c_7 + c_3/2)\|A\|_2 u \le (3/2)\gamma\|A\|_2 u, \end{aligned} \tag{5.14}$$

giving the remaining desired bounds for (5.12):

$$\begin{aligned} \|E\|_2 &\le \frac{3}{2}\gamma\|A\|_2 u.\frac{4}{3}.2uc_7\kappa.1 = 4c_7\gamma\|A\|_2 u^2\kappa \le c_7\|A\|_2 u, \\ \|\delta c\|_2 &\le 2\gamma^2(u + 2u\kappa)\|A\|_2\|b\|_2 u \le \gamma(1 + 2\gamma u)\|A\|_2\|b\|_2 u. \end{aligned}$$

Thus for problems whose condition satisfies (5.13), Algorithm 5.1 always produces $\bar{x}$ which is very close to $\tilde{x}$, see (5.8), and $\bar{y}$, where $\tilde{x}$ and $\bar{y}$ are the exact answers to a nearby problem (5.12), and so the algorithm is block strongly stable in the extended sense. Note that both E and δc are extremely small until $u\kappa$ approaches unity in magnitude. Thus for all but very large κ, this rounding error result is almost identical to that in (3.5), which suggests the algorithms will have very similar behavior (including the possibility of strongly stable use) until A is close to being rank deficient to working accuracy, i.e., $u\kappa \sim 1$. Then, in theory, the algorithm in Section 3 could be marginally superior.

6. Loss of backward stability of the original MGS. Now that we understand how to use MGS effectively, we are fully equipped to understand why many previous algorithms based on MGS were not backward stable for the ASF. Since we can now produce backward stable algorithms using Q_1 and R from MGS, we need not thoroughly analyze any previous algorithms based on Q_1 and R, but will illustrate the ideas in one fairly simple case.

Consider the following variant of (1.8):

$$(A, b) = (Q_1, h)\begin{pmatrix} R & d \\ 0 & 1 \end{pmatrix} \quad \text{by MGS,}$$

$$z = R^{-T}c, \qquad y = R^{-1}(d-z), \qquad x = b - Q_1(d-z). \tag{6.1}$$

The only difference between this and (1.8) is that d is computed as part of MGS rather than as $Q_1^T b$. Since this used to be the recommended approach for nonsingular systems $Ay = b$ and LLS via MGS [4], this is an algorithm that might be chosen in practice. The reason we consider it is that it is so close to our backward stable Algorithm 5.1. We see the *computed* $\bar{R}, \bar{h}, \bar{d}, \bar{z}, \bar{y}$ are identical in both, and only $\bar{x}$ differs, so we write $\bar{x}$ for the result computed by Algorithm 5.1, which is that in (5.2), and $\hat{x}$ for the result computed by (6.1). To simplify the analysis and bring out the essence clearly, we assume the computed $\bar{q}_k$ satisfy $\bar{q}_k^T\bar{q}_k = 1$, $k = 1, \ldots, n$, so each $\bar{P}_k$ is an *exactly* orthogonal matrix, and we assume there are no rounding errors in producing $\bar{z}$, $\bar{d}$, $\bar{h}$, $\bar{w}$, $\bar{x}$, $\bar{y}$ and $\hat{x}$ in their respective computations in (5.2) and (6.1):

$$\begin{aligned} \bar{z} &= \bar{R}^{-T}c, \qquad \begin{pmatrix} \bar{d} \\ \bar{h} \end{pmatrix} = \bar{P}^T\begin{pmatrix} 0 \\ b \end{pmatrix}, \qquad \begin{pmatrix} \bar{w} \\ \bar{x} \end{pmatrix} = \bar{P}\begin{pmatrix} \bar{z} \\ \bar{h} \end{pmatrix}, \\ \bar{y} &= \bar{R}^{-1}(\bar{d}-\bar{z}), \qquad \hat{x} = b - \bar{Q}_1(\bar{d}-\bar{z}), \end{aligned}$$

where $\bar{P} = \bar{P}_1 \cdots \bar{P}_n$ is an exactly orthogonal matrix. The only rounding errors we include are those that occur in the MGS reduction of A to give $\bar{Q}_1$ (and so $\bar{P}$) and $\bar{R}$. For these it follows from (5.3) that

$$\begin{pmatrix} E_7' \\ A + E_1' \end{pmatrix} = \begin{pmatrix} \bar{P}_{11} \\ \bar{P}_{21} \end{pmatrix}\bar{R}\,; \qquad \|E_i'\|_2 \le c_i' u \|A\|_2, \quad i = 1, 7,$$

so $\|\bar{P}_{11}\|_2 = \|E_7'\bar{R}^{-1}\|_2 \le c_7' u\kappa/[1-(c_1'+c_7')u\kappa] \le 2c_7' u\kappa$ if $4u\kappa \max\{c_1', c_7'\} \le 1$. Finally the theorem in [7] showed that

$$\bar{P}_{21} = \bar{Q}_1(I - \bar{P}_{11}).$$

We now use these to analyze this idealized old MGS algorithm. Since

$$\begin{pmatrix} \bar{w} \\ \bar{x} - b \end{pmatrix} = \bar{P}\begin{pmatrix} \bar{z} \\ \bar{h} \end{pmatrix} - \bar{P}\begin{pmatrix} \bar{d} \\ \bar{h} \end{pmatrix} = \bar{P}\begin{pmatrix} \bar{z} - \bar{d} \\ 0 \end{pmatrix},$$

we see

$$\begin{aligned} \bar{x} - b &= \bar{P}_{21}(\bar{z}-\bar{d}) = \bar{Q}_1(I - \bar{P}_{11})(\bar{z}-\bar{d}), \\ \bar{x} - \hat{x} &= -\bar{Q}_1\bar{P}_{11}(\bar{z}-\bar{d}) = -\bar{Q}_1 E_7'\bar{R}^{-1}(\bar{z}-\bar{d}) = \bar{Q}_1 E_7'\bar{y}. \end{aligned}$$

But then

$$\begin{aligned} \|\bar{Q}_1\|_2 &= \|\bar{P}_{21}(I - \bar{P}_{11})^{-1}\|_2 \le (1 - 2c_7' u\kappa)^{-1} \le 2, \\ \|\bar{x} - \hat{x}\|_2 &\le 2c_7' u \|A\|_2 \|\bar{y}\|_2, \end{aligned}$$

and $\|\bar{x}-\hat{x}\|_2$ has order of magnitude certainly no worse than the error in $\bar{x}$ in (2.9). Also $\bar{y}$ here is the same as $\bar{y}$ for an idealized version of the backward stable Algorithm 5.1, and so this idealized version of (6.1) is at least acceptable-error stable.

To see if it is backward stable, we examine the residuals. First we compare the size of the residual (2.6) corresponding to the top set of equations (2.1) for any backward stable method, with that for our idealized version of (6.1):

$$\begin{aligned}\|\hat{x}+A\bar{y}-b\|_2 &\leq \|\hat{x}-\bar{x}\|_2+\|\bar{x}+A\bar{y}-b\|_2\\ &\leq u\Big(c_b\|b\|_2+(c_A+2c_7')\|A\|_2\|\bar{y}\|_2\Big),\end{aligned}$$

to see that this residual is satisfactory. Unfortunately the residual corresponding to the bottom set of equations is not so good. For a backward stable algorithm we would have the bound in (2.7), while for the idealized version of (6.1)

$$\begin{aligned}\|A^T\hat{x}-c\|_2 &= \|A^T(\hat{x}-\bar{x})+A^T\bar{x}-c\|_2\\ &\leq 2c_7'u\|A\|_2^2\|\bar{y}\|_2+(c_c\|c\|_2+c_A\|A\|_2\|\bar{x}\|_2)u,\end{aligned}$$

where we see from the form of $A^T(\hat{x}-\bar{x})=-A^T\bar{Q}_1E_7'\bar{y}=-A^T\bar{Q}_1\bar{P}_{11}(\bar{d}-\bar{z})$ that we cannot expect any diminution of the $\|\bar{y}\|_2$ effect. In fact

$$\bar{y}=\bar{R}^{-1}(\bar{d}-\bar{z}),\qquad \|\bar{d}-\bar{z}\|_2^2=\|\bar{w}\|_2^2+\|\bar{x}-b\|_2^2$$

where $\bar{w}$ is ideally zero, so the term involving $\bar{y}$ can be about $c_7'u\kappa\|A\|_2\|\bar{x}-b\|_2$ in magnitude, and will give much too big a residual in ill-conditioned problems when $\|\bar{x}-b\|_2$ is not small. This means this idealized version of (6.1) is not backward stable, and so (6.1) is not either.

By choosing an algorithm that differs in only one simple computation from our backward stable algorithm of the previous section, we hope we have shown why many previous algorithms based on MGS are not backward stable, though they might be acceptable-error stable.

7. The corrected seminormal equations. If A is large and sparse it is often uneconomical to store and access the matrix Q, or even Q_1. Then, using the QR factorization, (1.5) can be rewritten as the *seminormal equations* (1.10). Paige [14] showed that using R from the Householder factorization in (1.10) with $b=0$ gives an acceptable-error stable algorithm for LUS. However, Björck [6] showed this is *not* true for LLS, that is for computing y with $c=0$ in (1.10).

Björck [6] also showed that for LLS the seminormal equations with one step of iterative refinement, called the *corrected seminormal equations* (CSNE), will usually give an acceptable-error stable result. Since R from MGS is numerically about as satisfactory as that obtained using the Householder QR factorization of A, the results of Paige and Björck regarding the seminormal equation method hold equally well for the QR factorization found via MGS, or Givens rotations. Here we extend some of these results to the full problem (1.1), that is, with neither b nor c zero in (1.1)–(1.3).

7.1. Analysis of seminormal equations. We now analyze the use of seminormal equations (SNE) with no refinement. First we give a brief analysis to show that this is not a backward stable method. Suppose we have $\bar{x}_1$ and $\bar{y}_1$ that satisfy

$$\bar{R}^T\bar{R}\bar{y}_1=A^Tb-c,\qquad \bar{x}_1=b-A\bar{y}_1$$

exactly, our equivalents of (1.10) if we use the computed $\bar{R}$ but assume no other errors, and consider the components of the residual for the ASF in (1.1) $\bar{r}_1\equiv b-\bar{x}_1-A\bar{y}_1=0$, $\bar{s}_1\equiv c-A^T\bar{x}_1$. Because $\bar{R}^T\bar{R}\neq A^TA$, $\bar{s}_1$ may not be zero. The residual of the top block corresponding to $\bar{x}$ and $\bar{y}$ computed by a backward stable method must satisfy the bound (2.6). The idealized SNE residual $\bar{r}_1=0$ certainly satisfies this. Similarly for a backward stable method the residual of the bottom block must satisfy the bound

$$\|c - A^T\bar{x}\|_2 \le u(c_A\|A\|_2\|\bar{x}\|_2 + c_c\|c\|_2), \tag{7.1}$$

see (2.7). We need to see if $\bar{s}_1 = c - A^T\bar{x}_1$ satisfies this. Assuming errors only in

$$A + E_0 = \tilde{Q}_1\bar{R}, \qquad \tilde{Q}_1^T\tilde{Q}_1 = I, \qquad \|E_0\|_2 \le c_0u\|A\|_2, \tag{7.2}$$

as for MGS, see (4.5), (or Householder and Givens), we get

$$\begin{aligned} \bar{s}_1 = c - A^T\bar{x}_1 = c - A^Tb + A^TA\bar{y}_1 &= (A^TA - \bar{R}^T\bar{R})\bar{y}_1 \\ &= (-\bar{R}^T\tilde{Q}_1^TE_0 - E_0^T\tilde{Q}_1\bar{R} + E_0^TE_0)\bar{y}_1. \end{aligned}$$

Hence, even when we neglect further rounding errors, the best we can say is

$$\|\bar{s}_1\|_2 \le c_0u[2(1 + c_0u) + c_0u]\|A\|_2^2\|\bar{y}_1\|_2. \tag{7.3}$$

If $\|A\|_2\|\bar{y}_1\|_2 \gg \|\bar{x}_1\|_2$ this residual is unnecessarily large compared with (7.1), showing that SNE certainly is not backward stable.

We now make a full rounding error analysis of SNE to show that it is not even acceptable-error stable. Here the c_i and E_i are not those in Sections 3 and 5. The computed $\bar{x}_1$ and $\bar{y}_1$ of (1.10) satisfy

$$(\bar{R} + E_3')^T(\bar{R} + E_4)\bar{y}_1 = -I_2[c - (A + E_2)^Tb], \tag{7.4}$$

$$\bar{x}_1 = I_1[b - (A + E_1')\bar{y}_1] = b - (A + E_1)\bar{y}_1 + e, \quad e \equiv D_1b, \tag{7.5}$$

where $I_j \equiv I + D_j$ is diagonal with $\|D_j\| \le u$, $j = 1, 2$. The rounding error effects (the I_1 and I_2) in the subtractions on the right hand sides can be taken into E_3' and partly intoE_1' giving E_3 and E_1, and we assume for here and later use

$$\|E_i\|_2 \le c_iu\|A\|_2, \quad i = 0, \dots, 6.$$

The true solution satisfies $A^TAy = A^Tb - c$, $x = b - Ay$, so we have

$$\bar{x}_1 - x = e + A(y - \bar{y}_1) - E_1\bar{y}_1. \tag{7.6}$$

Using $\tilde{Q}_1\bar{R} = A + E_0$, $\tilde{Q}_1^T\tilde{Q}_1 = I$, see for example (4.5), in (7.4) gives

$$\begin{aligned} (A + E_5)^T(A + E_6)\bar{y}_1 &= (A + E_2)^Tb - c, \\ E_5 \equiv E_0 + \tilde{Q}_1E_3, \qquad E_6 &\equiv E_0 + \tilde{Q}_1E_4. \end{aligned}$$

Subtracting $A^TAy = A^Tb - c$ and solving for $\bar{y}_1 - y$ we get

$$\bar{y}_1 - y = -(A^TA)^{-1}A^TE_6\bar{y}_1 + (A^TA)^{-1}\Big(E_2^Tb - E_5^TA\bar{y}_1 - E_5^TE_6\bar{y}_1\Big), \tag{7.7}$$

and taking norms

$$\|\bar{y}_1 - y\|_2 \le c_6u\kappa(1 + c_5u\kappa)\|\bar{y}_1\|_2 + u\sigma_n^{-1}\Big(\kappa c_2\|b\|_2 + \kappa c_5\|A\bar{y}_1\|_2\Big). \tag{7.8}$$

In comparison, a backward stable method gives at least as good as (see (2.15))

$$\|\bar{y} - y\|_2 \le 2u\kappa c_A\|\bar{y}\|_2 + u\sigma_n^{-1}\Big(2c_b\|b\|_2 + \kappa c_{c,A}\|b - A\bar{y}\|_2\Big).$$

With large κ this is much smaller than (7.8) if $\|b - A\bar{y}_1\|_2$ is small, but $\|b\|_2$ and so $\|A\bar{y}_1\|_2$ is large. The result (7.8) is however acceptable-error stable if $b = 0$, as was shown in [14]. Furthermore from (7.6) and (7.7)

$$\bar{x}_1 - x = e - E_1\bar{y}_1 + A(A^TA)^{-1}A^TE_6\bar{y}_1 - A(A^TA)^{-1}\Big(E_2^Tb - E_5^TA\bar{y}_1 - E_5^TE_6\bar{y}_1\Big),$$

so

$$\begin{aligned}\|\bar{x}_1 - x\|_2 &\leq u\|b\|_2 + u(c_1 + c_6)\|A\|_2\|\bar{y}_1\|_2 \\ &+ u\kappa\Big(c_2\|b\|_2 + c_5\|A\bar{y}_1\|_2 + c_5c_6u\|A\|_2\|\bar{y}_1\|_2\Big) \\ &\leq u(c_1 + c_6)\|A\|_2\|\bar{y}_1\|_2 + u\kappa(c_2'\|b\|_2 + c_5\|b - \bar{x}_1\|_2) + O(u^2),\end{aligned} \tag{7.9}$$

where $c_2' = c_2 + 1$. For a backward stable algorithm we have from (2.14)

$$\|\bar{x} - x\|_2 \leq uc_A\|A\|_2\|\bar{y}\|_2 + uc_b\|b\|_2 + u\kappa c_{c,A}\|\bar{x}\|_2.$$

We see that the error in $\bar{x}_1$ is fine if $b = 0$, i.e., the minimum norm solution as in [14], but if $\|b\|_2$ is large and $\|\bar{x}_1\|_2$ is not, then $\bar{x}_1$ is not a good solution. We conclude that SNE is neither acceptable-error nor backward stable for solving the ASF, as has already been shown in [6] for the special case LLS. This analysis nicely generalizes the two earlier analyses of [6, 14].

For the actual residual $\bar{s}_1$ here, $\bar{s}_1 \equiv c - A^T\bar{x}_1 = -A^T(\bar{x}_1 - x)$, so we get

$$\bar{s}_1 = -A^T(e - E_1\bar{y}_1) - A^TE_6\bar{y}_1 + \Big(E_2^Tb - E_5^TA\bar{y}_1 - E_5^TE_6\bar{y}_1\Big),$$

which gives, cf. (7.3),

$$\|\bar{s}_1\|_2 \leq u(c_1 + c_6)\|A\|_2^2\|\bar{y}_1\|_2 + u\|A\|_2\Big(c_2'\|b\|_2 + c_5\|b - \bar{x}_1\|_2\Big) + O(u^2),$$

which is again too large if $\|A\|_2\|\bar{y}_1\|_2 \gg \|\bar{x}_1\|_2$, see (7.1), and the method is not backward stable even if $b = 0$.

7.2. Iterative refinement. Since the solution of SNE on its own may not be satisfactory, we consider some ways in which iterative refinement may be used to improve this solution. Assume that, using the computed $\bar{R}$ from a QR factorization, we have obtained a solution x_1, y_1 for the (approximate) seminormal equations, see (1.10),

$$\bar{R}^T\bar{R}y = A^Tb - c, \qquad x = b - Ay. \tag{7.10}$$

We now want to find corrections δx_1, δy_1 so that $x = x_1 + \delta x_1$, $y = y_1 + \delta y_1$. If we let the residuals of the ASF (1.1) be

$$r_1 = b - x_1 - Ay_1, \qquad s_1 = c - A^Tx_1, \tag{7.11}$$

then δx_1, and δy_1 satisfy the linear system

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} \delta x_1 \\ \delta y_1 \end{pmatrix} = \begin{pmatrix} r_1 \\ s_1 \end{pmatrix}, \tag{7.12}$$

which again can be solved using the (approximate) seminormal equations

$$\bar{R}^T\bar{R}\delta y_1 = A^Tr_1 - s_1, \qquad \delta x_1 = r_1 - A\delta y_1. \tag{7.13}$$

This leads to our first iterative refinement algorithm:

ALGORITHM 7.1. **Iterative Refinement.** Set $x_0 = 0$, $y_0 = 0$, and for $i = 0, 1, 2, \ldots$ until convergence do

$$\begin{aligned} r_i &:= b - x_i - Ay_i, & s_i &:= c - A^T x_i, \\ \text{solve } R^T R\delta y_i &= A^T r_i - s_i, & \delta x_i &:= r_i - A\delta y_i, \\ x_{i+1} &:= x_i + \delta x_i, & y_{i+1} &:= y_i + \delta y_i. \end{aligned}$$

This algorithm requires two matrix-vector multiplications each with A and A^T. Note that for $i = 0$ we have $r_i = b$, $s_i = c$, so in the first step we solve the original seminormal equations.

For problem LLS we have $c = 0$ so $s_i = -A^T x_i$ and $A^T r_i - s_i = A^T(r_i + x_i)$. With $t_i \equiv r_i + x_i = b - Ay_i$ Algorithm 7.1 simplifies to

$$\begin{aligned} t_i &:= b - Ay_i, & \text{solve } R^T R\delta y_i &= A^T t_i, \\ y_{i+1} &:= y_i + \delta y_i. \end{aligned}$$

This is the variant that has been studied in [6] for the special case LLS. It only requires one matrix-vector multiplication each with A and A^T. We obtain a similar efficiency in the general case if we *set* $x_i = b - Ay_i$, so $r_i = 0$, then we have:

ALGORITHM 7.2. **Simplified Iterative Refinement I.** Set $x_0 = b$, $y_0 = 0$, and for $i = 0, 1, 2, \ldots$ until convergence do

$$\begin{aligned} s_i &:= c - A^T x_i, & \text{solve } R^T R\delta y_i &= -s_i, \\ y_{i+1} &:= y_i + \delta y_i, & x_{i+1} &:= b - Ay_{i+1}. \end{aligned}$$

This algorithm was apparently first suggested by W. Kahan, see [12]. It only requires one matrix-vector multiplication each with A and A^T. The first step, i.e. for $i = 0$, is identical to Algorithm 7.1. This algorithm gives acceptable-error stable results for general b and c. However, it is not as satisfactory as Algorithm 7.1. In particular the residual in the bottom block of equations can be much larger than for a backward stable algorithm.

In Algorithm 7.2 we have in theory $x_{i+1} = b - A(y_i + \delta y_i) = x_i - A\delta y_i$, so another simplified scheme is obtained by taking $x_{i+1} := x_i - A\delta y_i$ in Algorithm 7.2.

ALGORITHM 7.3. **Simplified Iterative Refinement II.** Set $x_0 = b$, $y_0 = 0$, and for $i = 0, 1, 2, \ldots$ until convergence do

$$\begin{aligned} s_i &:= c - A^T x_i, & \text{solve } R^T R\delta y_i &= -s_i, \\ y_{i+1} &:= y_i + \delta y_i, & x_{i+1} &:= x_i - A\delta y_i. \end{aligned}$$

For $i = 0$ this is again equivalent to Algorithm 7.1. This version is appropriate for LUS, when $b = 0$.

Algorithm 7.1 is the repeated application of the original solve. Hence the analysis of Section 7.1 for $\bar{x}_1$, $\bar{y}_1$ can be applied if the errors in computing the residuals are included (there are none in the original step). Since the detailed rounding error analysis is rather complex we give here a simplified analysis, where we only consider the error in $A + E_0 = \tilde{Q}_1 \bar{R}$, see (4.5). Eliminating r_i and s_i in Algorithm 7.1 we obtain

$$\begin{aligned} \bar{R}^T \bar{R}\delta y_i &= A^T(b - x_i - Ay_i) - c + A^T x_i = A^T A(y - y_i), \\ \delta x_i &= b - x_i - Ay_i - A\delta y_i. \end{aligned} \tag{7.14}$$

From (7.14) we have using $\delta y_i = y_{i+1} - y + y - y_i$

$$\bar{R}^T\bar{R}(y_{i+1} - y) = (\bar{R}^T\bar{R} - A^TA)(y_i - y) = \bar{R}^T M\bar{R}(y_i - y),$$

where $M \equiv \bar{R}^{-T}(\bar{R}^T\bar{R} - A^TA)\bar{R}^{-1}$ is symmetric, and using (7.2)

$$M = \bar{R}^{-T}E_0^T(A\bar{R}^{-1}) + (A\bar{R}^{-1})^T E_0\bar{R}^{-1} + (E_0\bar{R}^{-1})^T E_0\bar{R}^{-1}.$$

The smallest singular value of $\bar{R}$ satisfies

$$\sigma_n(\bar{R}) \geq \sigma_n - c_0 u\sigma_1 = \sigma_n(1 - c_0 u\kappa),$$

and so $\bar{R}$ is nonsingular if $c_0 u\kappa < 1$. With the assumption (5.13), and using $A\bar{R}^{-1} = \tilde{Q}_1 - E_0\bar{R}^{-1}$ we obtain the bounds

$$\|\bar{R}^{-1}\|_2 \leq \sigma_n^{-1}/(1 - c_0 u\kappa) < 4\sigma_n^{-1}/3, \qquad \|A\bar{R}^{-1}\|_2 < 4/3,$$

and the bound for the rate of convergence

$$\rho \equiv \|M\|_2 < 2 \cdot 16 c_0 u\kappa/9 + (4c_0 u\kappa/3)^2 < 4c_0 u\kappa. \tag{7.15}$$

This shows the important result that *the rate of convergence* (of $\bar{R}(y_i - y)$) *is proportional to* κu *and not* $\kappa^2 u$. Under the assumption (5.13) we have $\rho < 1$. Then $\bar{R}(y_{i+1} - y) \to 0$, so $y_{i+1} - y \to 0$, and from $x_{i+1} - x = -A(y_{i+1} - y)$ it follows that $x_{i+1} - x \to 0$ also.

In a practical iteration $y_i - y$ and and $x_i - x$ will only converge at best to an error level corresponding to a backward stable algorithm. However, even when the errors in computing the residuals are included, a bound roughly similar to (7.15) can be obtained for the *initial rate of convergence.* A description of this full error analysis is in preparation.

8. Conclusion. We have shown the augmented system formulation (ASF) with full column rank A

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}$$

can be effectively handled by various QR factorizations of A. In fact when Q_1 or $Q = (Q_1, Q_2)$ is available as well as R, we have shown there exist backward stable algorithms whether the QR factorization is based on Householder transformations, rotations, or the modified Gram-Schmidt (MGS) algorithm. The backward stability of all three approaches is about the same. We have also shown when these algorithms are effectively strongly stable in the extended sense of giving answers which are very close to the exact answers to a problem *of the exact same class* which is very close to the original problem. Also in this last case the MGS based algorithm has a very similar behavior to the others except possibly when $u\kappa$ is close to unity, where u is the floating point computer precision, and κ is the ratio of the largest to smallest singular values of A. The new stable MGS algorithm studied is that suggested in [7], and here we explained why previous MGS algorithms were not backward stable for the full ASF problem.

When no part of Q is available it is possible to solve the ASF problem via the seminormal equations, and in this case it makes no significant difference to the accuracy which one of the above three QR factorizations is used. We showed that the obvious use of the seminormal equations is not even acceptable-error stable (in that its solution errors can be significantly greater than those for a backward stable algorithm). However iterative refinement can be used to improve this. In a later paper we intend to show Algorithm 7.1 does give acceptable-error stability when $u\kappa \ll 1$. Moreover, when $u\kappa^2 \leq 1$, only one step of refinement is required to attain acceptable-error stability.

Acknowledgement. We thank Mario Arioli for some very useful comments, and for providing the nice diagram which helps to clarify some relationships and summarize the results.

LUS *Backward Stable Algorithms* *for computing x in (1.2)* *in extended sense:* *(1.9) using Householder or Givens*	**LLS** *Backward Stable Algorithms* *for computing y in (1.3)* *in extended sense:* *classic Householder or Givens*
$\Updownarrow$	$\Updownarrow$
ASF *Strongly Stable Algorithms* *for computing x in (1.1)* *in extended sense:* *(1.9) using Householder or Givens*	**ASF** *Strongly Stable Algorithms* *for computing y in (1.1)* *in extended sense:* *classic Householder or Givens*

ASF
Strongly Stable Algorithms

$\Downarrow$

ASF
Block Strongly Stable Algorithms:
(1.9) using Householder or Givens

$\Downarrow$

ASF
Block Strongly Stable Algorithms
in extended sense:
(1.9) using Householder or Givens
Algorithm 5.1 using MGS

$\Downarrow$

ASF
Backward Stable Algorithms
in extended sense

$\Downarrow$

ASF
Acceptable-error Stable Algorithms:
Classical MGS

$\Downarrow$

ASF
Weakly Stable Algorithms:
Seminormal Equations with Iterative Refinement
if $u\kappa \ll 1$

REFERENCES

[1] M. Arioli, I. S. Duff, and P. de Rijk, *On the augmented system approach to sparse least-squares problems*, Numer. Math., 55 (1989), pp. 667–684.

[2] M. Arioli and A. Laratta, *Error analysis of algorithms for computing the projection of a point onto a linear manifold*, Linear Algebra Appl., 82 (1986), pp. 1–26.

[3] ———, *Error analysis of an algorithm for solving an underdetermined linear system*, Numer. Math., 46 (1986), pp. 255–268.

[4] ———, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7 (1967), pp. 1–21.

[5] Å. Björck, *Iterative refinement of linear least squares solutions I*, BIT, 7 (1967), pp. 257–278.

[6] ———, *Stability analysis of the method of semi-normal equations for least squares problems*, Linear Algebra Appl., 88/89 (1987), pp. 31–48.

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

[8] J. R. Bunch, *The weak and strong stability of algorithms in numerical linear algebra*, Linear Algebra Appl., 88/89 (1987), pp. 49–66.

[9] J. R. Bunch, J. W. Demmel, and C. F. V. Loan, *The strong stability of algorithms for solving symmetric systems*, SIAM J. Matrix Anal. and Appl., 10 (1989), pp. 494–499.

[10] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, London and New York, 1981.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations. 2nd ed.*, Johns Hopkins Press, Baltimore, MD., 1989.

[12] G. H. Golub and J. H. Wilkinson, *Note on the Iterative Refinement of Least Squares Solution*, Numer. Math., 9 (1966), pp. 139–148.

[13] N. J. Higham, *The accuracy of solutions to triangular systems*, SIAM J. Numer. Anal., 26 (1989), pp. 1252–1265.

[14] C. C. Paige, *An error analysis of a method for solving matrix equations*, Math. Comp., 27 (1973), pp. 355–359.

[15] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[16] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, MA, 1986.

[17] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.