Journal of Computational Mathematics, Vol.25, No.5, 2007, 531–542.

# A QR DECOMPOSITION BASED SOLVER FOR THE LEAST SQUARES PROBLEMS FROM THE MINIMAL RESIDUAL METHOD FOR THE SYLVESTER EQUATION [*]

Zhongxiao Jia and Yuquan Sun
(*Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China*
*Email: jiazx@tsinghua.edu.cn, sunyq03@gmail.com*)

**Abstract**

Based on the generalized minimal residual (GMRES) principle, Hu and Reichel proposed a minimal residual algorithm for the Sylvester equation. The algorithm requires the solution of a structured least squares problem. They form the normal equations of the least squares problem and then solve it by a direct solver, so it is susceptible to instability. In this paper, by exploiting the special structure of the least squares problem and working on the problem directly, a numerically stable QR decomposition based algorithm is presented for the problem. The new algorithm is more stable than the normal equations algorithm of Hu and Reichel. Numerical experiments are reported to confirm the superior stability of the new algorithm.

*Mathematics subject classification:* 65F22, 65K10.
*Key words:* Least-squares solution, Preconditioning, Generalized singular value decomposition.

## 1. Introduction

Consider the numerical solution of the Sylvester equation

$$AX - XB = C, \tag{1.1}$$

where $A \in R^{n_1 \times n_1}, B \in R^{n_2 \times n_2}$ and $C \in R^{n_1 \times n_2}$ are given matrices, and $X$ is the solution matrix. Such kind of problems arise in various settings, and there are many methods for solving them [1, 2, 3, 4, 9, 10]. For large and sparse problems, based on the GMRES algorithm [8] for large unsymmetric linear systems, Hu and Reichel [6] present a minimal residual Krylov subspace method. In this case, a least squares problem of Kronecker product form must be solved, and a similar problem also sees [7]. In [6] they first form the normal equations system of the least squares problem and then solve it by a direct solver. Their algorithm is susceptible to instability and the computed solution may have poor accuracy due to the possibly high ill-conditioning of the normal equations system. In this paper we propose a new algorithm for solving the least squares problem. It is based on stable QR decompositions and fully exploit the special structure of the problem. Thus, it is more stable than a normal equations based solver. We also compare the cost of our algorithm with that of Hu and Reichel, showing that ours is a little bit more expensive than theirs but both are negligible, compared to the overall cost of the minimal residual method. This indicates that our improvement is significant.

In Section 2 we review the least squares problem to be solved in the minimal residual method. In Section 3 we show how to develop a stable QR decomposition based algorithm

[*] Received December 15, 2005; final revised September 1, 2006; accepted October 1, 2006.

for the resulting special least squares problem. Meanwhile, we compare the costs of our new algorithm and theirs. In Section 4 we report some numerical examples to show the stability and higher accuracy of our algorithm.

Some notations to be used in the paper are introduced. Denote by the superscript $T$ the transpose of a vector or a matrix, by $A^+$ the generalized inverse of $A$, by $\|\cdot\|_2$ the spectral norm, by $\kappa_2(A) = \|A\|_2\|A^+\|_2$ the condition number of $A$, by $\mathcal{R}^m$ the set of $m$-dimensional vectors, by $\mathcal{R}^{m\times n}$ the set of $m \times n$ matrices. Throughout the paper, $I$ is an identity with order clear in the context, $e_i$ is the $i$th row of $I$, and $\tilde{I}$ is the same as the identity matrix $I$ with an additional zero row.

## 2. The Least Squares Problem

The minimal residual method [6] given by Hu and Reichel replaces the subspace $\mathcal{K}_m(I \otimes A - B^T \otimes I, r_0)$ by another subspace of the form $\mathcal{K}_m(B^T, g) \otimes \mathcal{K}_m(A, f)$ for certain vectors $f$, $g \in R^n$. It is seen from Algorithm 5.2 of [6] that the construction of $f$, $g$ needs $\mathcal{O}(n_1n_2)$ flops. We should point out that usually the two subspaces are different. Hu and Reichel then use the Arnoldi process to generate the Hessenberg matrices $\tilde{H}_A$, $\tilde{H}_B$ and the orthonormal bases $W_m$ and $V_m$ of $\mathcal{K}_m(B^T, g)$ and $\mathcal{K}_m(A, f)$, respectively. The process uses $2n_1m^2$ flops + $m$ matrix $A$ by vector products and $2n_2m^2$ flops + $m$ matrix $B^T$ by vector products. Then one solves a least squares problem of the form

$$\min_{y_m \in \mathcal{R}^{m^2}} \|\tilde{r}_0 - (\tilde{I} \otimes \tilde{H}_A - \tilde{H}_B \otimes \tilde{I})y_m\|_2, \tag{2.1}$$

where $\tilde{I}$ is defined as before and $\tilde{H}_A, \tilde{H}_B$ are $(m+1) \times m$ Hessenberg matrices. Assume

$$H_A = Q_A R_A Q_A^T, \quad H_B = Q_B R_B Q_B^T$$

to be the Schur decompositions of $H_A$ and $H_B$, where $H_A$ and $H_B$ are the first $m$ rows of $\tilde{H}_A$ and $\tilde{H}_B$, and define the unitary matrices

$$\tilde{Q}_A = \begin{pmatrix} Q_A & \\ & 1 \end{pmatrix}, \quad \tilde{Q}_B = \begin{pmatrix} Q_B & \\ & 1 \end{pmatrix}$$

and the matrices

$$\tilde{R}_A = \tilde{Q}_A^T \tilde{H}_A Q_A, \quad \tilde{R}_B = \tilde{Q}_B^T \tilde{H}_B Q_B.$$

Then the leading $m \times m$ principal submatrix of $\tilde{R}_A$ is $R_A$, and the $(m+1)$st row of $\tilde{R}_A$ is given by

$$e_{m+1}^T \tilde{R}_A e_j = e_{m+1}^T \tilde{H}_A e_m e_m^T Q_A e_j, \quad 1 \le j \le m.$$

A similar result holds for $\tilde{R}_B$.

Let $r_0' = (\tilde{Q}_B \otimes \tilde{Q}_A)^T \tilde{r}_0$ and $y_m' = (Q_B \otimes Q_A)^T y_m$. Then (2.1) is equivalent to

$$\min_{y_m \in R^{m^2}} \|r_0' - (I_{m+1,m} \otimes \tilde{R}_A - \tilde{R}_B \otimes I_{m+1,m})y_m'\|_2. \tag{2.2}$$

The $(m+1)^2 \times m^2$ matrix in (2.2) has $m^2$ rows in common with the upper triangular matrix $R = (I \otimes R_A - R_B \otimes I)$. Let the remaining rows define the $(2m+1) \times m^2$ matrix $S$. Thus, for an appropriate permutation matrix $P$, (2.2) can be written as

$$\min_{y'_m \in \mathcal{R}^{m^2}} \left\| Pr'_0 - \begin{pmatrix} R \\ S \end{pmatrix} y'_m \right\|_2, \tag{2.3}$$

Hu and Reichel [6] determine the solution of this least squares problem by first solving the mathematically equivalent normal equations

$$\left(I + (SR^{-1})^T(SR^{-1})\right) y''_m = \left(I, (SR^{-1})^T\right) Pr'_0 \tag{2.4}$$

and then computing $y'_m$ from $Ry'_m = y''_m$.

The linear system (2.4) can be solved by an application of the Sherman-Morrison-Woodbury formula [5]. We now count flops (floating point operations) of each step for solving (2.4).

The first step is to construct the matrix $U = SR^{-1}$. The construction of $SR^{-1}$ is equivalent to the solution of the matrix equation

$$LU^T = S^T$$

for $U^T$, where $L = R^T$ is a $m^2 \times m^2$ lower triangular matrix. The matrix $L$ is a lower triangular matrix, whose diagonal blocks have the form $R_A + R_B(i,i)I$ and the non-blocks have the form $R_B(i,j)I$. Using the method in [5, p.91], we can solve the linear equation $Lu_j^T = s_j^T$ with about $\frac{3}{2}m^3$ flops, where $u_j^T$ and $s_j^T$ are the $j$th columns of $U^T$ and $S^T$. Therefore, the flops for solving all the $2m$ columns of $S^T$ is about $3m^4$ without taking the structure of $S^T$ into account. It is seen from above that the matrix $U = SR^{-1}$ can be computed using fewer than $3m^4$ flops.

Now (2.4) becomes

$$\left(I + U^TU\right) y''_m = \left(I, U^T\right) Pr'_0. \tag{2.5}$$

In the second step we get $y''_m$ with an application of the Sherman-Morrison-Woodbury formula:

$$y''_m = \left(I + U^T(I + UU^T)^{-1}U\right)\left(I, U^T\right) Pr'_0. \tag{2.6}$$

Since $U$ is a $(2m+1) \times m^2$ matrix, it costs $8m^4$ flops to construct the matrix $UU^T$ without considering its structure, and the inverse matrix in (2.6) is $(2m+1) \times (2m+1)$ instead of $m^2 \times m^2$. The cost of the Cholesky factorization of $(I + UU^T)$ is $\frac{1}{3}(2m+1)^3$ flops. Then we can get the matrix $X = (I + (UU^T))^{-1}U$ using roughly $8m^4$ flops. By the special structure of $U$, the matrix multiplication $U^TX$ costs

$$2m^2 \cdot m \cdot m^2 + 2m \sum_{k=1}^{m} k \cdot m^2 \approx 3m^5$$

flops and the rest cost of (2.6) is approximately $m^4$ flops. So the total cost of (2.6) is approximately $3m^5$ flops and the total cost of the normal equations solver is $3m^5$ flops. Recall the cost of the Arnoldi process for (1.1). We see that if matrix by vector products are not dominant and $m^3$ is of order $\max\{n_1, n_2\}$ then the cost of the normal equations solver is comparable to that of the Arnoldi process.

Note that relative error of the computed solution $\tilde{x}_{LS}$ to a general least squares problem $\min \|b - Ax\|_2$ by a normal equations solver is a moderate multiple of $\kappa_2(A)^2\epsilon = \|A\|_2^2\|A^+\|_2^2\epsilon$, while the relative error of the computed solution obtained by a stable QR decomposition for solving the LS problem directly is a moderate multiple of

$$\left(\kappa_2(A) + \frac{\rho_{LS}}{\|b\|}\kappa_2(A)^2\right)\epsilon, \tag{2.7}$$

where

$$\rho_{LS} = \|(I - AA^{+})b\|_2 \tag{2.8}$$

and $\epsilon$ is the machine precision; see [5]. It is clearly seen that a normal equations solver may produce a much poorer computed solution than the QR decomposition solver does when $\rho_{LS}/\|b\|$ is small and $\kappa_2(A)$ is large. Only when $\rho_{LS}/\|b\| \approx 1$ both solvers have roughly the same condition numbers and may produce computed solutions with comparable accuracy. However, even in this worst case, a direct based Cholesky factorization solver may break down when $\kappa_2(A) = \mathcal{O}(1/\sqrt{\epsilon})$ but a direct LS QR decomposition based solver can only do so when $\kappa_2(A) = \mathcal{O}(1/\epsilon)$. So generally a stable LS solver should be preferable for accuracy and robustness. In the next section, by exploiting the special structure of the problem under consideration, we develop a direct LS solver based on a sequence of stable QR decompositions and present a stable algorithm for solving (2.1), whose cost is comparable to that of the normal equations solver of Hu and Reichel.

## 3. A QR Decomposition Based Solver

Without considering the structure of (2.1), if it were solved by a standard QR decomposition directly, the cost would be $\frac{4}{3}m^6$ flops [5, p. 263]. This way would destroy the special block $\tilde{H}^{B}_{i,j}\tilde{I}$, and the cost would be basically $\frac{1}{4}m$ times of that of the previous normal equations solver. We now propose a stable QR decomposition based solver by fully exploiting the special structure of problem (2.1), so that we can get a more accurate solution. We also show that the cost of our algorithm is comparable to that of Hu and Reichel.

The $(m+1)^2 \times m^2$ coefficient matrix $K = \tilde{I} \otimes \tilde{H}_A - \tilde{H}_B \otimes \tilde{I}$ in the least squares problem (2.1) has the structure

$$K = \begin{pmatrix} \tilde{H}_A - H^{(B)}_{1,1}\tilde{I} & -H^{(B)}_{1,2}\tilde{I} & \cdots & -H^{(B)}_{1,m}\tilde{I} \\ -H^{(B)}_{2,1}\tilde{I} & \tilde{H}_A - H^{(B)}_{22}\tilde{I} & \cdots & -H^{(B)}_{2,m}\tilde{I} \\ & -H^{(B)}_{3,2}\tilde{I} & & \\ & & \ddots & \vdots \\ & & -H^{(B)}_{m,m-1}\tilde{I} & \tilde{H}_A - H^{(B)}_{m,m}\tilde{I} \\ & & & -H^{(B)}_{m+1,m}\tilde{I} \end{pmatrix},$$

which is a structured matrix and is in block upper Hessenberg form. We can fully make use of the special structure of $K$ to develop a stable QR decomposition of it and then solve the least squares problem (2.1) by the QR decomposition in a standard way.

Write $K^{(0)} = K$ and $\tilde{r}^{(0)}_0 = \tilde{r}_0$. The first step we do is to annihilate the last row of $\tilde{H}_A - H^{(B)}_{11}\tilde{I}$ and $-H^{(B)}_{21}\tilde{I}$ in $K$ so that the first block column of $K$ is in upper triangular form. We proceed as follows: Let

$$\begin{pmatrix} \tilde{H}_A - H^{(B)}_{1,1}\tilde{I} \\ -H^{(B)}_{2,1}\tilde{I} \end{pmatrix} = Q_1 R_1 = Q_1 \begin{pmatrix} \tilde{R}_1 \\ 0 \end{pmatrix}$$

be a full QR decomposition of the first block column of $K$, where $Q_1$ is an $2(m+1) \times 2(m+1)$ orthogonal matrix and $\tilde{R}_1$ is an $m \times m$ upper triangular matrix. Define the $(m+1)^2 \times (m+1)^2$ orthogonal matrix

$$\tilde{Q}_1 = \begin{pmatrix} Q_1^T & \\ & I \end{pmatrix}$$

and premultiply $K^{(0)}$ and $\tilde{r}_0$ by $\tilde{Q}_1$. This transformation only changes the first two block rows of $K^{(0)}$ and the first $2m+2$ entries of $\tilde{r}_0$. Then the transformed matrix is

$$K^{(1)} = \tilde{Q}_1 K^{(0)} = \begin{pmatrix} \tilde{R}_1 & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ & -H_{3,2}^{(B)}\tilde{I} & \ddots & \vdots \\ & & \ddots & \tilde{H}_A - H_{m,m}^{(B)}\tilde{I} \\ & & & -H_{m+1,m}^{(B)}\tilde{I} \end{pmatrix}$$

and the transformed vector is $\tilde{Q}_1 \tilde{r}_0^{(0)} = \tilde{r}_0^{(1)}$. Next we deal with the second block column of $K^{(1)}$, which has $3m+3$ rows and is of the form

$$\begin{pmatrix} \times & \times & \cdots & \times \\ k_{m+1,m+1}^{(1)} & k_{m+1,m+2}^{(1)} & \cdots & k_{m+1,2m}^{(1)} \\ \multicolumn{4}{c}{Q_1^T(:, m+2:2m+2)(\tilde{H}_A - H_{2,2}^{(B)}\tilde{I})} \\ \multicolumn{4}{c}{-H_{3,2}^{(B)}\tilde{I}} \end{pmatrix} \begin{pmatrix} \times \ \times \ \cdots \ \times \\ B_2 \end{pmatrix}, \tag{3.1}$$

in which $Q_1^T(:, m+2:2m+2)$ denotes the matrix consisting of the $(m+2)$-nd to the $(2m+2)$-nd columns of $Q_1^T$ and the $\times$ block row on both sides has $m$ rows. Now we only need to deal with the last $2m+3$ rows

$$B_2 = \begin{pmatrix} k_{m+1,m+1}^{(1)} \ k_{m+1,m+2}^{(1)} \ \cdots \ k_{m+1,2m}^{(1)} \\ Q_1^T(:, m+2:2m+2)(\tilde{H}_A - H_{2,2}^{(B)}\tilde{I}) \\ -H_{3,2}^{(B)}\tilde{I} \end{pmatrix}.$$

Suppose

$$B_2 = Q_2 R_2 = Q_2 \begin{pmatrix} \tilde{R}_2 \\ 0 \end{pmatrix}$$

be a full QR decomposition of $B_2$, where $Q_2$ is an $(2m+3)\times(2m+3)$ orthogonal matrix and $\tilde{R}_2$ is an $m \times m$ upper triangular matrix. Define the $(m+1)^2 \times (m+1)^2$ orthogonal matrix $\tilde{Q}_2$ by

$$\tilde{Q}_2 = \begin{pmatrix} I_1 & & \\ & Q_2^T & \\ & & I_2 \end{pmatrix},$$

where $I_1$ is the identity matrix of order $m$ and $I_2$ the identity matrix of order $(m+1)^2-(3m+3)$. Then it is seen that in $K^{(2)} = \tilde{Q}_2 K^{(1)}$ and $\tilde{r}_0^{(2)} = \tilde{Q}_2 \tilde{r}_0^{(1)}$ only the $(m+1)$-th to the $(3m+3)$-th rows of $K^{(1)}$ and $\tilde{r}_0^{(1)}$ are premultiplied by $Q_2^T$ and the rest rows remain unchanged. Therefore, the first two block rows and columns of $K^{(2)}$ are already in upper triangular form.

At the $i$-th step we already have the matrices $K^{(i-1)}$ and $\tilde{Q}_{i-1}$. The first $i-1$ block columns of $K^{(i-1)}$ is already in upper triangular form

$$\begin{pmatrix} \tilde{R}_1 & \times & \cdots & \times \\ & \tilde{R}_2 & \cdots & \times \\ & & \ddots & \vdots \\ & & & \tilde{R}_{i-1} \\ & & & 0 \end{pmatrix}$$

and the $i$-th block column is

$$\begin{pmatrix} \times \\ \star \\ \hat{Q}_{i-1}^T(:,(m+1)(i-1)+1:(m+1)i)(\tilde{H}_A - H_{i,i}^{(B)}\tilde{I}) \\ -H_{i+1,i}^{(B)}\tilde{I} \\ 0 \end{pmatrix} \begin{pmatrix} \times \\ B_i \\ 0 \end{pmatrix},$$

where the $\times$ block row has $m(i-1)$ rows and the $\star$ block row has $i-1$ rows. So $B_i$ is $2(m+1)+(i-1)$ by $m$. We make a full QR decomposition

$$B_i = Q_i R_i = Q_i \begin{pmatrix} \tilde{R}_i \\ 0 \end{pmatrix},$$

where $Q_i$ is an $(2(m+1)+i-1) \times (2(m+1)+i-1)$ orthogonal matrix and $\tilde{R}_2$ is an $m \times m$ upper triangular matrix.

Now we form the $(m+1)^2 \times (m+1)^2$ orthogonal matrix $\tilde{Q}_i$ by

$$\tilde{Q}_i = \begin{pmatrix} I_1 & & \\ & Q_i^T & \\ & & I_2 \end{pmatrix},$$

where $I_1$ is the identity matrix of order $m(i-1)$ and $I_2$ the identity matrix of order $(m+1)^2 - (3m+i+1)$.

Premultiply $K^{(i-1)}$ and $\tilde{r}_0^{(i-1)}$ by $\tilde{Q}_i$ to define $K^{(i)}$ and $\tilde{r}_0^{(i)}$, in which only the $m(i-1)+1$-th to the $(m+1)(i+1)$-th rows of $K^{(i-1)}$ and $\tilde{r}_0^{(i-1)}$ are premultiplied by $Q_i^T$ and the rest rows remain unchanged. Then the first $i$ block columns of $K^{(i)}$ will be in upper triangular form. At the end, at the $m$-th step we compute a full QR decomposition of the last $3m+1$ rows of the $m$-th block column of $K^{(m-1)}$ and get $Q_m$ and $\tilde{R}_m$. We then define $\tilde{Q}_m$ and form $K^{(m)} = \tilde{Q}_m K^{(m-1)}$ and $\tilde{r}_0^{(m)} = \tilde{Q}_m \tilde{r}_0^{(m-1)}$.

The above procedure can be written as

$$\tilde{Q}_m \tilde{Q}_{m-1} \cdots \tilde{Q}_1 K = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{R}_1 & \times & \times & \times \\ & \tilde{R}2 & \times & \times \\ & & \ddots & \vdots \\ & & & \tilde{R}_m \\ & & & 0 \end{pmatrix} \tag{3.2}$$

and $\tilde{Q}_m \tilde{Q}_{m-1} \cdots \tilde{Q}_1 \tilde{r}_0 \tilde{r}_0^{(m)}$. Define the $(m+1)^2 \times (m+1)^2$ orthogonal matrix $Q = (\tilde{Q}_{m-1} \cdots \tilde{Q}_1)^T$. Consequently, we achieve a QR decomposition:

$$K = Q \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}. \tag{3.3}$$

Now the structured least squares problem (2.1) reduces to

$$\min_{y_m \in \mathcal{R}^{m^2}} \left\| \tilde{r}_0^{(m)} - \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} y_m \right\|_2. \tag{3.4}$$

This can be solved by a conventional back substitution.

Two comments are in order. First, we obtain a stable QR decomposition of $K$ if we compute a stable QR decomposition at each step in the above process. This can be done if we use Householder transformations to achieve a QR decomposition. Second, it is *unnecessary* to explicitly form the orthogonal matrix $Q$. We only need to update $\tilde{r}_0$ step by step. So we can save a lot of computational cost. Therefore, we have now proposed a stable QR decomposition based solver for (2.1).

Now we count the computational cost of our new solver. As described above, there are the following two main sources of the cost:

1. **The cost of the total QR decomposition**

   Note that for an $m \times n$ matrix the cost of the full QR decomposition using Householder transformations is $2n^2(m - \frac{n}{3})$ without explicitly forming $Q$ [5, p. 225]. Then the total cost of our solver for the QR decomposition is
$$\sum_{i=1}^{m} 2m^2(2m + i + 1 - \frac{m}{3}) \approx \frac{13}{3}m^4.$$

2. **The cost of successively getting $\tilde{R}$ and $\tilde{r}_0^{(m)}$**

   The cost of premultiplying an $n \times q$ matrix by $r$ Householder transformations is $2qr(2n-r)$ flops [5, p. 213]. At the $i$-th step, we have $r = m$, $n = 2m + i + 1$, $q = m(m - i) + 1$. So the total cost of successively getting $\tilde{R}$ and $\tilde{r}_0^{(m)}$ is
$$\begin{aligned}&\sum_{i=1}^{m-1} 2m(m^2 - mi + 1)(3m + 2i + 2) + 2m(5m + 2)\\ &= 2m \sum_{i=1}^{m-1} [3m^3 - m^2 i - 2mi^2 + 2m^2 - 2mi + 3m + 2i + 2] + 10m^2 + 4m\\ &= \tfrac{11}{3}m^5 - m^4 + \mathcal{O}(m^3).\end{aligned}$$

The least squares problem (3.4) can be solved using only $m^4$ flops. Therefore, the cost of the minimal solution by our QR decomposition based solver is basically $\frac{11}{3}m^5$. Recall that the normal equations solver of Hu and Reichel costs $3m^5$ flops. So our algorithm is a little bit more expensive but is comparable to theirs. Recall the cost of the Arnoldi process for (1.1). We see that if matrix by vector products are not dominant and $m^3$ is of order $\max\{n_1, n_2\}$ then the cost of our solver is comparable to that of the Arnoldi process, similar to the normal equations solver.

## 4. Numerical Experiments

In this section we report on some numerical results to show the sharp stability of our new algorithm. All the experiments presented in this section were performed on an Intel Pentium(R) IV with CPU 2.8 GHz and 512 MB RAM using Matlab 7.0.1.

In order to compare the accuracy and stability of the two algorithms, we need to know the exact solution of (2.1). At the same time, we need to know $\tilde{r}_0$. Unlike the linear system $Ax = b$, we could trivially construct the *unique* right-hand side $b$ by $b = Ax^*$ once the exact solution

Table 4.1: Condition numbers and relative errors.

| $\frac{\rho_{LS}}{\|\tilde{r}_0\|}$ | The new solver | | | The normal equations solver | | |
|---|---|---|---|---|---|---|
| | residual | CPU-time | total CPU time | residual | CPU-time | total CPU time |
| 4.75e-13 | 4.21e-8 | 0.08 | 2.31 | 2.81 | 0.06 | 2.30 |
| 4.75e-8 | 3.42e-6 | 0.08 | 2.30 | 1.72 | 0.05 | 2.29 |
| 4.75e-5 | 3.40e-3 | 0.08 | 2.29 | 1.74 | 0.05 | 2.29 |

Table 4.2: Condition numbers of $\tilde{H}_A$,$\tilde{H}_B$ and $K$.

| $k,\ j$ | $k=2, j=2$ | $k=4, j=4$ | $k=6, j=6$ | $k=8, j=8$ | $k=10, j=10$ |
|---|---|---|---|---|---|
| $\kappa(\tilde{H}_A)$ | 1.00e2 | 1.24e4 | 1.00e6 | 1.00e8 | 9.83e9 |
| $\kappa(\tilde{H}_B)$ | 1.06e3 | 1.01e4 | 1.43e3 | 7.5e2 | 1.51e2 |
| $\kappa(K)$ | 1.10e2 | 1.00e4 | 1.00e6 | 1.00e8 | 9.80e9 |

$x^*$ is given. For the least squares problem $\min_{y_m \in \mathcal{R}^{m^2}} \|\tilde{r}_0 - Ky_m\|_2$, the situation becomes nontrivial and complicated. Assume that the vector $y^*$ is the exact solution. Then we have $\tilde{r}_0 = r + Ky^*$, where $r$ is the residual and satisfies $K^T r = 0$, that is, $r \perp Ky^*$. Therefore, given $K$ and $y^*$, there are *infinitely many* right-hand sides $\tilde{r}_0$ such that $\min_{y_m \in \mathcal{R}^{m^2}} \|\tilde{r}_0 - Ky_m\|_2$ have the same exact solution $y^*$.

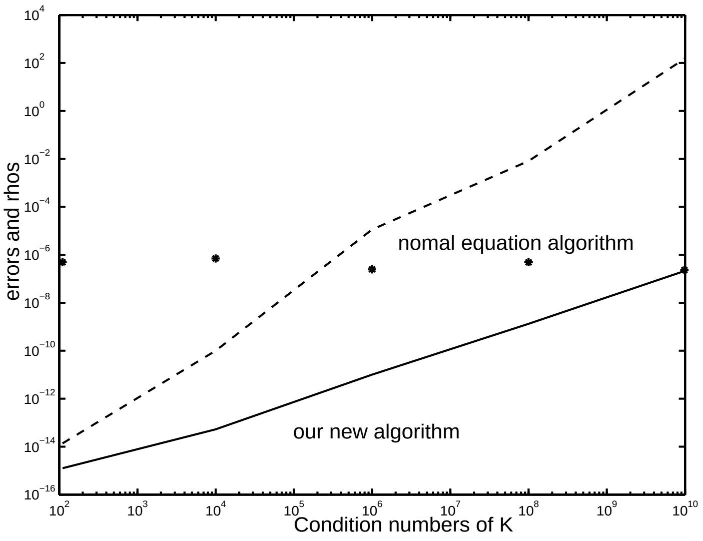
Given $y^*$, we can construct various $\tilde{r}_0$ in such a way: Choose a vector $c$ and orthogonalize it against the columns of $K$ by the Gram-Schmidt procedure with iterative refinements. Then the resulting vector is orthogonal to the columns of $K$ to working precision. We then just take it to be $r$. In this and the following examples, we take all $y^* = (1, 1, \ldots, 1)^T$ and $c = (1, \ldots, 1)^T$. Furthermore, since $\|r\|_2 = \rho_{LS}$, we can vary $\rho_{LS}$ by scaling $r$ and $y^*$.

**Example 1**. We first tested a problem similar to Example 7.3 in [6], where the matrices $A$, $B$ and $C$ are $63 \times 63$ and the maximal Arnoldi steps $m = 20$. Here we constructed the same type matrices, but they were $3000 \times 3000$. We took $m = 20$ and generated the orthonormal bases $V_{m+1}$ and $W_{m+1}$ of $\mathcal{K}_{m+1}(A, f)$ and $\mathcal{K}_{m+1}(B^T, g)$, respectively, as done in [6]. At the same time, we got two $(m+1) \times m$ matrices $\tilde{H}_A$ and $\tilde{H}_B$. Here note that if we constructed the right-hand vector $\tilde{r}_0$ from matrix $C$, we would have to construct the matrix $W_m \otimes V_m$ of order $3000^2 \times 20^2$, which would use 3600MB main memory and be too large to be stored. So instead we constructed the vectors $y^*$ and $\tilde{r}_0$ in the way described above. Additionally, we computed the condition number of $K$.

For this example, we should keep in mind that (i) the flops of the normal equations solver and our solver are both comparable to the Arnoldi process for $m = 20$ and (ii) the cost of constructing $f$ and $g$ is $\mathcal{O}(n^2)$ flops, several times bigger than that of the Arnoldi process for $m = 20$. So the construction of $f$ and $g$ dominated the CPU timings of the two algorithms, as also seen from experiments.

Table 4.1 shows the behavior of the two algorithms, where the CPU-time denoted the CPU timings of the whole algorithms in which the least squares problems were solved either by the normal equations solver of Hu and Reichel [6] or by the QR decomposition based solver proposed and the total CPU time denoted the total CPU timings for solving the large Sylvester equation (1.1) by the algorithm of Hu and Reichel and our modified variant. Since the condition number of $K$ is $3.84 \times 10^9$, the normal equations solver delivers solutions with no accuracy, as expected. The normal equations solver uses a little less CPU-time, but the new solver gives

(a)



(b)

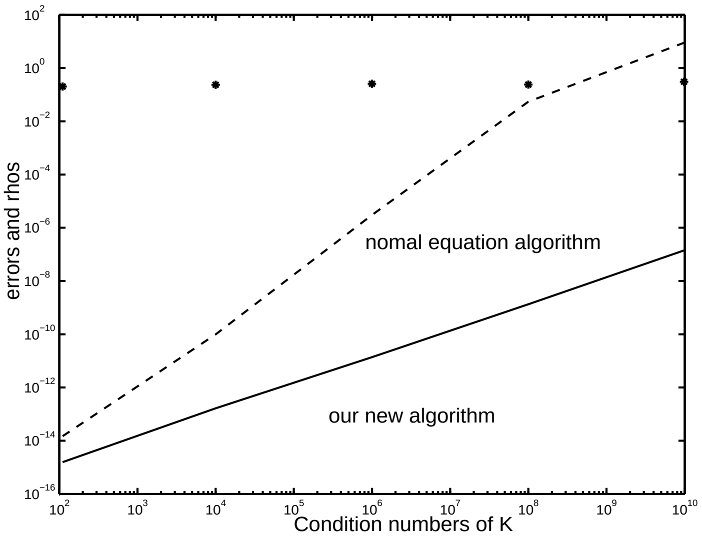


Fig. 4.1. Example 2: the relative errors with different condition numbers and $\rho_{LS}/\|\tilde{r}_0\|$. (a) All $\rho_{LS}/\|\tilde{r}_0\|$ are $\mathcal{O}(e^{-7})$, and (b) All $\rho_{LS}/\|\tilde{r}_0\|$ are $\mathcal{O}(e^{-1})$.

much more accurate solutions. Furthermore, the CPU timings consumed by the two algorithms for solving (1.1) were almost equal, while the CPU timings used by the two solvers for the least squares problem both were negligible, compared with the total CPU timings in each algorithm for solving (1.1).

The new solver gave poorer solutions as $\rho_{LS}/\|\tilde{r}_0\|$ became bigger. This was consistent with our previous analysis. The numerical experiments shows that although the new solver is a little bit more expensive than the normal equations solver, the cost of both solvers were negligible compared to the overall cost of the whole algorithm. So our solver is significant as it can compute solutions to much more accuracy than the normal equations solver did.

**Example 2**. Since we are only concerned with solving the least squares problem (2.1) and compare accuracy and stability of the normal equations solver and our new solver, it suffices to construct Hessenberg matrices $\tilde{H}_A$ and $\tilde{H}_B$ themselves. Let $D_1 = diag(1 : 9, 10^j)$ and

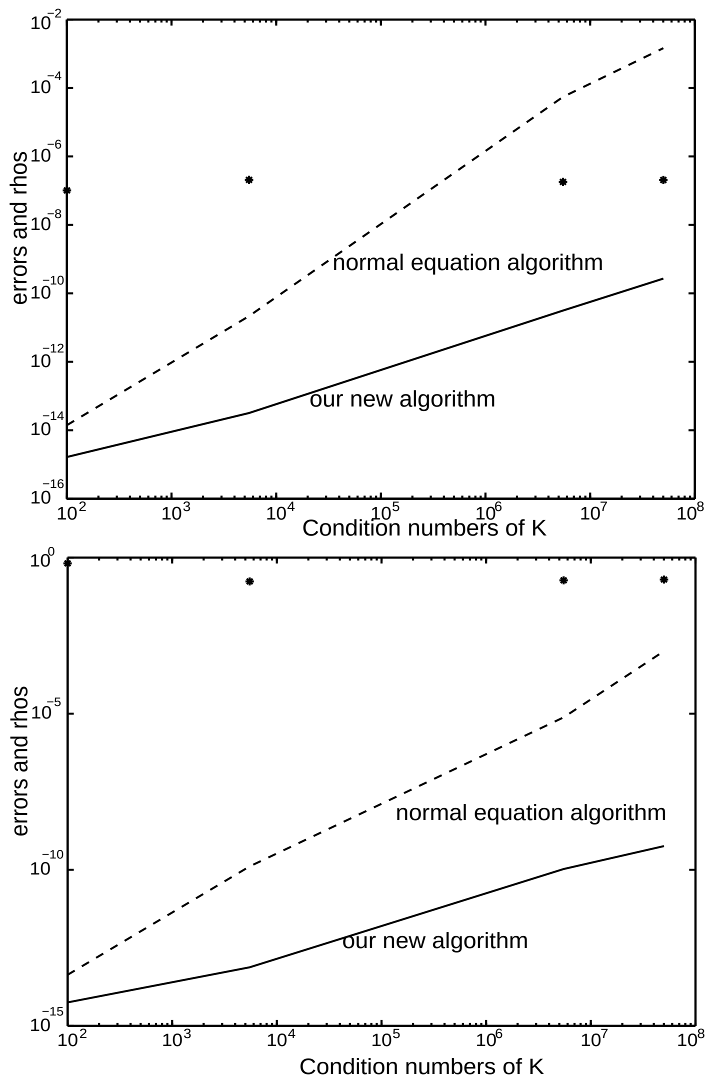


Fig. 4.2. Example 3: the relative errors with different $\kappa(K)$. (a) All $\rho_{LS}/\|\tilde{r}_0\|$ are $\mathcal{O}(e^{-7})$, and (b) All $\rho_{LS}/\|\tilde{r}_0\|$ are $\mathcal{O}(e^{-1})$.

$D_2 = diag(10^{-k}, -9 : -1)$. Here we adapted Matlab notations. Then we first used a random orthogonal matrix $Q$ to get two full matrices $\tilde{D}_1 = Q^T D_1 Q, \tilde{D}_2 = Q^T D_2 Q$ and transformed $\tilde{D}_1$ and $\tilde{D}_2$ to the Hessenberg matrices $H_A$ and $H_B$ by orthogonal similarity transformations. We then added an appropriate row to $H_A$ and $H_B$ to get two $11 \times 10$ Hessenberg matrices $\tilde{H}_A = [H_A; zeros(1,9), 10]$ and $\tilde{H}_B = [H_B; zeros(1,9), 10]$ in the standard Matlab format. Then the matrix $K = (\tilde{I} \otimes \tilde{H}_A - \tilde{H}_B \otimes \tilde{I})$ of the least squares problem was $121 \times 100$. We can vary $j$, $k$ to get different $D_1$, $D_2$ and thus different $K$.

Table 4.2 shows the condition numbers of $\tilde{H}_A$, $\tilde{H}_B$ and $K$, and Table 4.3 lists the relative errors. Fig. 4.1 shows the behavior of the two algorithms with different condition numbers and $\rho_{LS}/\|\tilde{r}_0\|$, in which the points "stars" are the values of $\rho_{LS}/\|\tilde{r}_0\|$.

It is observed from both Table 4.3 and Fig. 4.1 that as the condition numbers of $K$ became bigger both algorithms obtained poorer computed solutions. But the relative errors obtained by our algorithm were much smaller than those by the normal equations solver. In particular, if

Table 4.3: Relative errors with different $\rho_{LS}/\|\tilde{r}_0\|$'s.

| | | | | | | |
|---|---|---|---|---|---|---|
| | $\rho_{LS}/\|\tilde{r}_0\|$ | 0.24 | 0.92 | $1-10^{-4}$ | $1-10^{-10}$ | 1 |
| $\kappa(K)$ | The normal equations solver | 3.75e-10 | 2.02e-10 | 1.18e-8 | 2.44e-6 | 3.79e-4 |
| $=1e4$ | The new solver | 7.31e-14 | 1.17e-12 | 1.92e-11 | 4.26e-9 | 6.01e-6 |
| | $\rho_{LS}/\|\tilde{r}_0\|$ | 0.35 | 0.92 | $1-10^{-4}$ | $1-10^{-9}$ | 1 |
| $\kappa(K)$ | The normal equations solver | 3.00e-3 | 4.74e-2 | 1.84e-2 | 98.10 | 1.41e4 |
| $=8e8$ | The new solver | 7.60e-10 | 1.78e-8 | 1.23e-7 | 1.01e-5 | 3.67e-3 |

Table 4.4: Condition numbers of $\tilde{H}_A$, $\tilde{H}_B$ and $K$.

| $k,\ j$ | $k=2, j=2$ | $k=4, j=3$ | $k=6, j=7$ | $k=8, j=5$ |
|---|---|---|---|---|
| $\kappa(\tilde{H}_A)$ | 1.11e2 | 1.00e4 | 1.00e6 | 1.00e8 |
| $\kappa(\tilde{H}_B)$ | 1.10e2 | 1.00e3 | 1.00e7 | 1.00e5 |
| $\kappa(K)$ | 1.00e2 | 5.50e3 | 5.55e6 | 5.00e7 |

Table 4.5: Relative errors with different $\rho_{LS}/\|\tilde{r}_0\|$'s.

| | | | | | |
|---|---|---|---|---|---|
| | $\rho_{LS}/\|\tilde{r}_0\|$ | 0.19 | $1-10^{-4}$ | $1-1^{-10}$ | 1 |
| $\kappa(K)$ | The normal equations solver | 5.36e-11 | 8.33e-10 | 1.18e-6 | 1.78e-4 |
| $=5.5e3$ | The new solver | 9.46e-14 | 6.85e-12 | 6.11e-9 | 9.82e-6 |
| | $\rho_{LS}/\|\tilde{r}_0\|$ | 0.17 | 0.93 | $1-10^{-6}$ | $1-10^{-11}$ |
| $\kappa(K)$ | The normal equations solver | 2.57e-3 | 1.11e-3 | 0.67 | 1.08e3 |
| $=5e7$ | The new solver | 6.06e-10 | 7.37e-9 | 9.90e-7 | 6.90e-4 |

the condition number of $K$ reached $10^8$, then the normal equations solver delivered a very poor computed solution; even worse, if $\kappa(K) \approx 10^{10}$, the computed solution had no accuracy at all. Actually, the relative errors of the computed solutions by the normal equations solver increased by a multiple of $\kappa^2(K)$. In contrast, our algorithm got much more accurate computed solutions. All these confirm our previous analysis and indicate the sharp stability of our algorithm.

Fig. 4.1 also shows the influence of $\rho_{LS}/\|\tilde{r}_0\|$. We observe from Fig. 4.1 that the accuracy of computed solutions by our algorithm became a little poorer as $\rho_{LS}/\|\tilde{r}_0\|$ grew but was not very near one. This illustrates that the term $\rho_{LS}/\|\tilde{r}_0\|\kappa^2(K)$ played a role in the computed accuracy but the effect was not considerable when $\rho_{LS}/\|\tilde{r}_0\|$ was far from zero but not near one. However, this term was sudden to have had strong influence when $\rho_{LS}/\|\tilde{r}_0\|$ very near one, as was seen from Table 4.3. This indicates that the bound of (2.7) was very conservative when $\rho_{LS}/\|\tilde{r}_0\|$ was not near one and became quite sharp when it was very near one. On the other hand, we found from Table 4.3 that our new algorithm was considerably more stable than the normal equations solver even when $\rho_{LS}/\|\tilde{r}_0\|$ was near one.

**Example 3.** In this example we let $D_1 = diag(1:19, 10^k)$ and $D_2 = diag(-10^j, -19:-1)$. As for example 1, we construct two $21 \times 20$ matrices $\tilde{H}_A$, $\tilde{H}_B$. Then the matrix $K = (\tilde{I} \otimes \tilde{H}_A - \tilde{H}_B \otimes \tilde{I})$ of the least squares problem is $441 \times 400$.

Table 4.4 gives condition numbers of $\tilde{H}_A$, $\tilde{H}_B$ and $K$. Table 4.5 shows the influence of $\rho_{LS}/\|\tilde{r}_0\|$. Fig. 4.2 describes the behavior of the two algorithms.

Fig. 4.2 clearly shows that the relative errors of the computed solutions obtained by the two solvers became bigger when the condition numbers were increasingly bigger. The relative errors of the computed solutions by the direct normal equations solver were proportional to $\kappa^2(K)$, while those of our new algorithm grew linearly with respect to $\kappa(K)$ if $\rho_{LS}/\|\tilde{r}_0\|$ was not close to one. Particularly, as $\kappa(K)$ was as large as $10^9$, the direct normal equations solver delivered a solution with no accuracy, but the computed solution by our new solver was satisfactory. If $\rho_{LS}/\|\tilde{r}_0\|$ was very close to one, the relative errors obtained by our solver were found proportional to $\kappa^2(K)$. They were, however, much smaller than those obtained by the normal equations solver.

**Acknowledgement.** We thank the anonymous referee for his/her suggestions and comments which improved the presentation of this paper. This project supported by the National Natural Science Foundation of China (No. 10471074) and the Doctoral Program of the Ministry of Education, China.

## References

[1] R. Bartels and G.W. Stewart, Algorithm 432: Solution of the matrix equation $AX + XB = C$, *Commun. ACM*, **15** (1972), 820-826.

[2] G. Birkhoff, R.S. Varga and D.M. Young, Alternating direction implicit methods, in Advances in Computing, Vol. 3, Academic Press, New York, 1962, 189-273.

[3] N.S. Ellner and E.L. Wachspress, Alternating direction implicit iteration for systems with complex spectra, *SIAM J. Numer. Anal.*, **28** (1991), 859-870.

[4] G.H. Golub, C.F. Nash and C.F. Van Loan, A Hessenberg-Schur method for problem $AX+XB = C$, *IEEE T. Automat. Control*, **AC-24** (1979), 909-913.

[5] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, Baltimore and London, 1996.

[6] D.Y. Hu and L. Reichel, Krylov subspace methods for the Sylvester equation, *Linear Algebra Appl.*, **172** (1992), 283-314.

[7] M. Robbé and M. Sadkane, A convergence analysis of GMRES and FOM methods for Sylvester equations, *Numer. Algorithms*, **30** (2002), 71-89.

[8] Y. Saad and M.H. Schultz, GMRES: A generalized minimal residua algorithm for solviing non symmetric linear systems, *SIAM J. Sci. Stat. Comput.*, **7** (1986), 856-869.

[9] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, 1962.

[10] E.L. Wachspress, Iterative solution of the Lyapunov matrix equation, *Appl. Math. Lett.*, **1** (1988), 87-90.