SIAM J. MATRIX ANAL. APPL.
Vol. 24, No. 1, pp. 292–301

PRODUCT TRIANGULAR SYSTEMS WITH SHIFT*

CARLA D. MORAVITZ MARTIN† AND CHARLES F. VAN LOAN‡

Abstract. Systems of the form $(R^{(1)} \cdots R^{(p)} - \lambda I)x = b$, where each $R^{(i)}$ is an n-by-n upper triangular matrix, can be solved in $O(pn^3)$ flops if the matrix of coefficients is explicitly formed. We develop a new method for this system that circumvents the explicit product and requires only $O(pn^2)$ flops to execute. The error bounds for the new algorithm are essentially the same as the error bounds for the explicit method. The new algorithm extends readily to the situation when $R^{(1)}$ is upper quasi-triangular.

Key words. back-substitution, matrix products

AMS subject classification. 65F05

PII. S0895479801396051

1. Introduction.

Suppose the matrices $R^{(1)}, R^{(2)}, \ldots, R^{(p)} \in \mathbb{R}^{n\times n}$ are all upper triangular and that we want to solve

$$(R^{(1)} \cdots R^{(p)} - \lambda I)x = b, \tag{1.1}$$

where $\lambda \in \mathbb{R}$, $b \in \mathbb{R}^n$, and the matrix of coefficients is nonsingular. This problem arises in various product eigenvalue problems $(A^{(1)} \cdots A^{(p)})x = \lambda x$. (See [2].) In these settings the A-matrices are reduced to triangular form without the explicit formation of the product. The computation of eigenvectors by back-substitution involves the solution of a product triangular system with shift.

One way to solve (1.1) is to form the upper triangular matrix $(R^{(1)} \cdots R^{(p)} - \lambda I)$ and then use back-substitution. We refer to this as the *explicit method* and note that it is an $O(pn^3)$ procedure because of the matrix-matrix multiplications. In this paper we develop an *implicit method* that carefully engages selected parts of the coefficient matrix during the back-substitution process. The implicit algorithm requires only $O(pn^2)$ flops and has the same backward error properties as the explicit method. Our contribution therefore adds to the set of product-free matrix algorithms that have recently been developed for problems that involve matrix products. See [1] for an overview of this important paradigm and [4] for an example.

To illustrate the main idea without getting bogged down in details we first work through the $p = 2$ case. We then discuss the general algorithm, including a simple extension that can handle the case when $R^{(1)}$ is upper quasi-triangular. An error analysis and some reaffirming numerical results complete the paper.

2. The $p = 2$ case.

Consider the situation when the product in (1.1) involves just two matrices. Assume that $S, T \in \mathbb{R}^{n\times n}$ are upper triangular and that the system

$$(ST - \lambda I)x = b, \qquad \lambda \in \mathbb{R},\ b \in \mathbb{R}^n,$$

*Received by the editors October 3, 2001; accepted for publication (in revised form) by I. C. F. Ipsen February 15, 2002; published electronically July 9, 2002. This work was supported by NSF grant CCR-9901988.

http://www.siam.org/journals/simax/24-1/39605.html

†Center for Applied Mathematics, Cornell University, 657 Rhodes Hall, Ithaca, NY 14853-7510 (carlam@cam.cornell.edu).

‡Department of Computer Science, Cornell University, 4130 Upson Hall, Ithaca, NY 14853-7510 (cv@cs.cornell.edu).

is nonsingular. Suppose $1 \le k \le n-1$. Define

$$\begin{aligned} S_+ &= S(n-k{:}n, n-k{:}n), \\ T_+ &= T(n-k{:}n, n-k{:}n), \\ x_+ &= x(n-k{:}n), \\ b_+ &= b(n-k{:}n) \end{aligned}$$

and observe that

$$(S_+T_+ - \lambda I_{k+1})\, x_+ = b_+$$

is just the trailing $(k+1)$-by-$(k+1)$ portion of $(ST-\lambda I)x = b$. It has the form

$$\left(\begin{bmatrix} \sigma & u^T \\ 0 & S_c \end{bmatrix} \begin{bmatrix} \tau & v^T \\ 0 & T_c \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda I_k \end{bmatrix} \right) \begin{bmatrix} \gamma \\ x_c \end{bmatrix} = \begin{bmatrix} \beta \\ b_c \end{bmatrix},$$

where $\sigma, \tau, \gamma, \beta \in \mathbb{R}$, $u, v, x_c, b_c \in \mathbb{R}^k$, and $S_c, T_c \in \mathbb{R}^{k\times k}$. The two rows in this equation tell us that

$$(S_cT_c - \lambda I_k)x_c = b_c \tag{2.1}$$

and

$$\gamma = \frac{\beta - \sigma v^T x_c - u^T T_c x_c}{\sigma\tau - \lambda}\,. \tag{2.2}$$

The efficiency of "ordinary" back-substitution relies on the fact that at the start of step k the vector x_c is available and that the scalar γ can be obtained in $O(k)$ flops. However, in our product system if we literally use (2.2) to compute γ, then $O(k^2)$ flops are required because of the matrix-vector product T_cx_c. Unless this computation can be rearranged we are headed for an overall algorithm that needs $O(n^3)$ flops.

Fortunately there is a way to do this through a simple recursion that can be used to compute $w_+ = T_+x_+$ (the "next" w) from $w_c = T_cx_c$ (the "current" w). Since

$$w_+ = T_+x_+ = \begin{bmatrix} \tau & v^T \\ 0 & T_c \end{bmatrix} \begin{bmatrix} \gamma \\ x_c \end{bmatrix} = \begin{bmatrix} \tau\gamma + v^Tx_c \\ w_c \end{bmatrix}$$

it follows that we need only compute the scalar $\omega \equiv \tau\gamma + v^Tx_c$ to get w_+ from w_c. Thus, we can carry out each of the transitions $x_c \to x_+$ and $w_c \to w_+$ in $O(k)$ flops, and this renders the following overall procedure.

IMPLICIT METHOD ($p = 2$).

$x_c \leftarrow b_n/(S(n,n)T(n,n) - \lambda)$

$w_c \leftarrow T(n,n)x_c$

for $k = 1{:}n-1$

$\sigma \leftarrow S(n-k, n-k)$; $u \leftarrow S(n-k, n-k+1{:}n)^T$

$\tau \leftarrow T(n-k, n-k)$; $v \leftarrow T(n-k, n-k+1{:}n)^T$

$\gamma \leftarrow (\beta - \sigma(v^Tx_c) - u^Tw_c)/(\sigma\tau - \lambda)$

$\omega \leftarrow \tau\gamma + v^Tx_c$

$x_c \leftarrow \begin{bmatrix} \gamma \\ x_c \end{bmatrix}$; $w_c \leftarrow \begin{bmatrix} \omega \\ w_c \end{bmatrix}$

end

$x \leftarrow x_c$

In step k there are two length-k inner products, i.e., $v^T x_c$ and $u^T w_c$. Thus, the algorithm requires a total of $2n^2$ flops.

3. The general case. We now extend the above algorithm to the case when the coefficient matrix involves the product of p upper triangular matrices:

$$\left(R^{(1)} \cdots R^{(p)} - \lambda I\right) x = b. \tag{3.1}$$

Suppose $1 \le k \le n-1$. Define

$$\begin{aligned} R_+^{(i)} &= R^{(i)}(n-k{:}n, n-k{:}n), \qquad i = 1{:}p, \\ x_+ &= x(n-k{:}n), \\ b_+ &= b(n-k{:}n) \end{aligned}$$

and observe that

$$\left(R_+^{(1)} \cdots R_+^{(p)} - \lambda I_{k+1}\right) x_+ = b_+$$

is just the trailing $(k+1)$-by-$(k+1)$ portion of (3.1). It has the form

$$\left(\begin{bmatrix} \sigma_1 & u_1^T \\ 0 & R_c^{(1)} \end{bmatrix} \cdots \begin{bmatrix} \sigma_p & u_p^T \\ 0 & R_c^{(p)} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda I_k \end{bmatrix} \right) \begin{bmatrix} \gamma \\ x_c \end{bmatrix} = \begin{bmatrix} \beta \\ b_c \end{bmatrix}, \tag{3.2}$$

where $\sigma_i, \gamma, \beta \in \mathbb{R}$, $u_i, x_c, b_c \in \mathbb{R}^k$, and $R_c^{(i)} \in \mathbb{R}^{k \times k}$ for $i = 1{:}p$. In order to develop the necessary recursions for the back-substitution process we need to look more carefully at the product of the partitioned triangular matrices in this equation. It is easy to show by induction on p that

$$\begin{bmatrix} \sigma_1 & u_1^T \\ 0 & R_c^{(1)} \end{bmatrix} \cdots \begin{bmatrix} \sigma_p & u_p^T \\ 0 & R_c^{(p)} \end{bmatrix} = \begin{bmatrix} \sigma_1 \cdots \sigma_p & \displaystyle\sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1}) u_j^T R_c^{(j+1)} \cdots R_c^{(p)} \\ 0 & R_c^{(1)} \cdots R_c^{(p)} \end{bmatrix}.$$

By substituting this into (3.2) we conclude that

$$(R_c^{(1)} \cdots R_c^{(p)} - \lambda I_k) x_c = b_c \tag{3.3}$$

and

$$\gamma = \frac{\beta - \sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1})\, u_j^T\, w_c^{(j)}}{\sigma_1 \cdots \sigma_p - \lambda}, \tag{3.4}$$

where $w_c^{(p)} = x_c$ and $w_c^{(j)} = R_c^{(j+1)} \cdots R_c^{(p)} x_c$ for $j = p-1{:}-1{:}1$. In order to effect an $O(k)$ transition from x_c to x_+ we need to develop $O(k)$ update recipes for the w-vectors. In particular, we need a fast method for computing

$$w_+^{(j)} = R_+^{(j+1)} \cdots R_+^{(p)} x_+, \qquad j = 1{:}p-1, \tag{3.5}$$

assuming that we are in possession of $w_c^{(1)}, \ldots, w_c^{(p)}$. Since the matrices

$$R_+^{(i)} = \begin{bmatrix} \sigma_i & u_i^T \\ 0 & R_c^{(i)} \end{bmatrix}, \qquad i = j{+}1{:}p,$$

are upper triangular it follows that each $w_+^{(j)}$ has the form

$$w_+^{(j)} \;=\; \begin{bmatrix} \omega_j \\ w_c^{(j)} \end{bmatrix},$$

and so we just need a quick way to compute the scalars $\omega_1, \ldots, \omega_p$. Since $w_+^{(p)} = x_+$ we have

$$\begin{bmatrix} \omega_p \\ w_c^{(p)} \end{bmatrix} \;=\; \begin{bmatrix} \gamma \\ x_c \end{bmatrix},$$

and so $\omega_p = \gamma$. Simple formulae for $\omega_{p-1}, \ldots, \omega_1$ can be derived from (3.5). This equation tells us that $w_+^{(j)} = R_+^{(j+1)} w_+^{(j+1)}$, i.e.,

$$\begin{bmatrix} \omega_j \\ w_c^{(j)} \end{bmatrix} \;=\; \begin{bmatrix} \sigma_{j+1} & u_{j+1}^T \\ 0 & R_c^{(j+1)} \end{bmatrix} \begin{bmatrix} \omega_{j+1} \\ w_c^{(j+1)} \end{bmatrix}.$$

Thus,

$$\omega_j \;=\; \sigma_{j+1}\omega_{j+1} + u_{j+1}^T w_c^{(j+1)}$$

for $j = p{-}1{:}\,{-}1{:}1$. Combining all this we obtain the following procedure.

IMPLICIT METHOD (GENERAL p).

$x_c \;\leftarrow\; b_n/(R^{(1)}(n,n)\cdots R^{(p)}(n,n) - \lambda)$
$w_c^{(p)} = x_c$
$w_c^{(j)} \;\leftarrow\; R_c^{(j+1)}(n,n) w_c^{(j+1)} \qquad (j = p{-}1{:}\,{-}1{:}1)$
for $k = 1{:}n-1$
  $\sigma_j \;\leftarrow\; R^{(j)}(n-k, n-k) \qquad (j = 1{:}p)$
  $u_j \;\leftarrow\; R^{(j)}(n-k, n-k+1{:}n)^T \qquad (j = 1{:}p)$
  $\beta \;\leftarrow\; b(n-k)$
  $$\gamma \;=\; \left(\beta \;-\; \sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1})\; u_j^T\; w_c^{(j)} \right) \Big/ \; (\sigma_1 \cdots \sigma_p \;-\; \lambda)$$
  if $k < n-1$
    $\omega_p \;\leftarrow\; \gamma;\; w_c^{(p)} \leftarrow x_c$
    $\omega_j \leftarrow \sigma_{j+1}\omega_{j+1} + u_{j+1}^T w_c^{(j+1)} \qquad (j = p{-}1{:}\,{-}1{:}1)$
    $w_c^{(j)} \;\leftarrow\; \begin{bmatrix} \omega_j \\ w_c^{(j)} \end{bmatrix} \qquad (j = 1{:}p)$
  end
  $x_c \;\leftarrow\; \begin{bmatrix} \gamma \\ x_c \end{bmatrix}$
end
$x \leftarrow x_c$

There are p length-k inner products to compute in step k, i.e., $u_j^T w_c^{(j)}$, $j = 1{:}p$. Thus, the overall algorithm requires pn^2 flops.

4. The quasi-triangular case. In the eigenvector application mentioned in the introduction it is sometimes the case that $R^{(1)}$ is upper quasi-triangular, i.e., block upper triangular with 1-by-1 and 2-by-2 blocks along the diagonal. The 2-by-2 bumps correspond to complex conjugate eigenvalue pairs.

The implicit algorithm generalizes in a straightforward way to handle this situation. To see how to carry out a step that corresponds to a 2-by-2 bump we rewrite (3.2) as follows:

$$\left(\begin{bmatrix} S_1 & U_1^T \\ 0 & R_c^{(1)} \end{bmatrix} \cdots \begin{bmatrix} S_p & U_p^T \\ 0 & R_c^{(p)} \end{bmatrix} - \begin{bmatrix} \lambda I_2 & 0 \\ 0 & \lambda I_k \end{bmatrix} \right) \begin{bmatrix} \gamma \\ x_c \end{bmatrix} = \begin{bmatrix} \beta \\ b_c \end{bmatrix}, \tag{4.1}$$

where $S_i \in \mathbb{R}^{2\times 2}$, $U_i \in \mathbb{R}^{(n-k)\times 2}$, $R_c^{(i)} \in \mathbb{R}^{k\times k}$, $\beta \in \mathbb{R}^2$, $b_c \in \mathbb{R}^k$, and $x_c \in \mathbb{R}^k$ are given. Our goal is to compute efficiently $\gamma \in \mathbb{R}^2$. Following the corresponding discussion in section 3 it can be shown that

$$\begin{bmatrix} S_1 & U_1^T \\ 0 & R_c^{(1)} \end{bmatrix} \cdots \begin{bmatrix} S_p & U_p^T \\ 0 & R_c^{(p)} \end{bmatrix} = \begin{bmatrix} S_1 \cdots S_p & \displaystyle\sum_{j=1}^{p} (S_1 \cdots S_{j-1}) U_j^T R_c^{(j+1)} \cdots R_c^{(p)} \\ 0 & R_c^{(1)} \cdots R_c^{(p)} \end{bmatrix}.$$

From this it follows that $(R_c^{(1)} \cdots R_c^{(p)} - \lambda I_k) x_c = b_c$ and

$$(S_1 \cdots S_p - \lambda I_2)\, \gamma = \beta - \sum_{j=1}^{p} (S_1 \cdots S_{j-1})\, U_j^T\, w_c^{(j)}, \tag{4.2}$$

where $w_c^{(p)} = x_c$ and

$$w_c^{(j)} = R_c^{(j+1)} \cdots R_c^{(p)} x_c, \qquad j = p{-}1{:}\,{-}1{:}1. \tag{4.3}$$

Thus, the next two components of x, i.e., $\gamma = x(n-k-1{:}n-k)$, are found by solving (4.2), a 2-by-2 linear system. The update of the w-vectors is analogous to the update derived in section 3 for the triangular case. Since $w_+^{(p)} = x_+$ we have

$$w_+^{(p)} \equiv \begin{bmatrix} \omega_p \\ w_c^{(p)} \end{bmatrix} = \begin{bmatrix} \gamma \\ x_c \end{bmatrix},$$

and so $\omega_p = \gamma$. From (4.3) $w_+^{(j)} = R_+^{(j+1)} w_+^{(j+1)}$, i.e.,

$$\begin{bmatrix} \omega_j \\ w_c^{(j)} \end{bmatrix} = \begin{bmatrix} S_{j+1} & U_{j+1}^T \\ 0 & R_c^{(j+1)} \end{bmatrix} \begin{bmatrix} \omega_{j+1} \\ w_c^{(j+1)} \end{bmatrix},$$

and so the vectors $\omega_j \in \mathbb{R}^2$ can be found via

$$\omega_j = S_{j+1}\omega_{j+1} + U_{j+1}^T w_c^{(j+1)}$$

for $j = p{-}1{:}\,{-}1{:}1$. Thus, the transition from $\{x_c, w_c^{(1)}, \ldots, w_c^{(p)}\}$ to $\{x_+, w_+^{(1)}, \ldots, w_+^{(p)}\}$ involves $O(k)$ flops even if a 2-by-2 bump is encountered.

5. Backward error analysis. We show that backward error analyses for the explicit and implicit methods are essentially the same. The goal is not to derive the "best possible" results but simply to substantiate observed numerical behavior. In particular, we show that both the explicit and implicit methods produce a computed solution $\hat{x}$ that solves a "nearby" system

$$(R^{(1)} \cdots R^{(p)} - \lambda I + E)\hat{x} \;=\; b, \tag{5.1}$$

where the perturbation matrix E satisfies

$$\| E \| \;=\; O\left(\mathbf{u}\left(\| R^{(1)} \| \cdots \| R^{(p)} \| \;+\; |\lambda|\right)\right) \tag{5.2}$$

with $\mathbf{u}$ being the unit roundoff and $\| \cdot \|$ designating (say) the 2-norm.

For simplicity we assume that $R^{(1)}$ is upper triangular. The analysis for the quasi-triangular case is similar and basically yields the same results.

Consider the explicit method first. It begins with the computation of the matrix of coefficients $A = R^{(1)} \cdots R^{(p)} - \lambda I$:

$A_1 = R^{(1)}$
for $j = 2{:}p$
    $A_j = \text{fl}(A_{j-1}R^{(j)})$
end
$\hat{A} = \text{fl}(A_p - \lambda I)$

Here, $\text{fl}(x \text{ op } y)$ is the floating point version of x op y, where x and y are floating point scalars, vectors, or matrices and "op" is some legitimate operation between them. Applying standard floating point error results that can be found in [2] or [3], it can be shown that

$$A_j \;=\; A_{j-1}R^{(j)} + E_j, \qquad\qquad \| E_j \| \;=\; O\left(\mathbf{u} \| A_{j-1} \| \| R^{(j)} \|\right) \tag{5.3}$$

for $j = 2{:}p$. Taking into account the roundoff error associated with the λ-shift gives

$$\hat{A} \;=\; A_p \;-\; \lambda I \;+\; F_1, \qquad\qquad \| F_1 \| \;=\; O\left(\mathbf{u}\left(\| A_p \| + |\lambda|\right)\right), \tag{5.4}$$

and so by a simple inductive argument we find that

$$\hat{A} \;=\; R^{(1)} \cdots R^{(p)} - \lambda I \;+\; F_1 \;+\; \sum_{j=2}^{p} E_j R^{(j+1)} \cdots R^{(p)}.$$

At this point back-substitution is applied to $\hat{A}x = b$ and produces an $\hat{x}$ that satisfies

$$(\hat{A} + F_2)\hat{x} = b, \qquad\qquad \| F_2 \| \;=\; O\left(\mathbf{u}\| \hat{A} \|\right). \tag{5.5}$$

Combining all of these results, it is not hard to show that (5.1) holds with

$$E = F_1 + \sum_{j=2}^{p} E_j R^{(j+1)} \cdots R^{(p)} + F_2.$$

Taking norms in this equation and simplifying the right-hand side with (5.3), (5.4), and (5.5) confirms (5.2).

To show that (5.1) and (5.2) apply to the implicit method, we proceed by induction on n. The $n = 1$ case holds because the $\hat{x}$ produced by the implicit method is identical to the $\hat{x}$ that is produced by the explicit method.

Assume that $1 \leq k \leq n-1$. Using the notation of section 3 and "hats" to designate computed quantities, the induction argument is complete if we can show that

$$\left(R_+^{(1)} \cdots R_+^{(p)} \; - \; \lambda I_{k+1} \; + \; E_+\right) \hat{x}_+ = b_+ \tag{5.6}$$

with

$$\| \, E_+ \, \| \; = \; O\left(\mathbf{u}\left(\| \, R_+^{(1)} \, \| \cdots \| \, R_+^{(p)} \, \| \; + \; |\lambda|\right)\right), \tag{5.7}$$

given that

$$\left(R_c^{(1)} \cdots R_c^{(p)} \; - \; \lambda I_k \; + \; E_c\right) \hat{x}_c = b_c \tag{5.8}$$

with

$$\| \, E_c \, \| \; = \; O\left(\mathbf{u}\left(\| \, R_c^{(1)} \, \| \cdots \| \, R_c^{(p)} \, \| \; + \; |\lambda|\right)\right). \tag{5.9}$$

To that end partition (5.6)

$$E_+ \; = \; \begin{bmatrix} \epsilon & e^T \\ 0 & E_c \end{bmatrix}$$

conformably with (3.2).

From (5.6) we see that our task is to show that $\hat{\gamma}$ satisfies

$$\left(\begin{bmatrix} \sigma_1 & u_1^T \\ 0 & R_c^{(1)} \end{bmatrix} \cdots \begin{bmatrix} \sigma_p & u_p^T \\ 0 & R_c^{(p)} \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda I_k \end{bmatrix} + \begin{bmatrix} \epsilon & e^T \\ 0 & E_c \end{bmatrix}\right) \begin{bmatrix} \hat{\gamma} \\ \hat{x}_c \end{bmatrix}$$

$$= \begin{bmatrix} \sigma_1 \cdots \sigma_p - \lambda + \epsilon & \displaystyle\sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1}) u_j^T R_c^{(j+1)} \cdots R_c^{(p)} + e^T \\ 0 & R_c^{(1)} \cdots R_c^{(p)} - \lambda I_k + E_c \end{bmatrix} \begin{bmatrix} \hat{\gamma} \\ \hat{x}_c \end{bmatrix} = \begin{bmatrix} \beta \\ b_c \end{bmatrix}$$

(5.10)

with

$$|\epsilon| \; = \; O\left(\mathbf{u}\left(\| \, R_+^{(1)} \, \| \cdots \| \, R_+^{(p)} \, \| \; + \; |\lambda|\right)\right) \tag{5.11}$$

and

$$\| \, e \, \| \; = \; O\left(\mathbf{u}\left(\| \, R_+^{(1)} \, \| \cdots \| \, R_+^{(p)} \, \| \; + \; |\lambda|\right)\right). \tag{5.12}$$

Using elementary properties of the 2-norm, it can be shown that (5.7) follows from (5.9), (5.11), and (5.12).

Before we set out to verify (5.11) and (5.12) we establish a handy tilde notation that can be used to indicate accuracy to machine precision. If M is a matrix, then $\widetilde{M}$ is an approximation that satisfies $\| M - \widetilde{M} \|/\| M \| = O(\mathbf{u})$. The notation is a useful way to account for the rounding errors in floating point matrix-vector multiplication. Indeed, if M is a floating point matrix and v is a floating point vector, then $\mathrm{fl}(Mv) = \widetilde{M}v$.

When we account for all the rounding errors associated with the evaluation of the right-hand side in (3.4) we find that

$$\hat{\gamma} \;=\; \frac{\beta \;-\; \sum_{j=1}^{p} \left((\sigma_1 \cdots \sigma_{j-1})(1+\mu_j) \tilde{u}_j^T \hat{w}_c^{(j)} \right)}{\sigma_1 \cdots \sigma_p (1+\delta_1) \;-\; \lambda(1+\delta_2)}, \tag{5.13}$$

where $|\delta_i| = O(\mathbf{u})$ for $i = 1, 2$ and $|\mu_j| = O(\mathbf{u})$ for $i = 1{:}p$. We shall establish below that the computed w_c vectors satisfy

$$\hat{w}_c^{(j)} = (R_c^{(j+1)} + F_c^{(j+1)}) \cdots (R_c^{(p)} + F_c^{(p)}) \hat{x}_c, \qquad \| F_c^{(j)} \| \;=\; O\left(\mathbf{u} \| R_c^{(j)} \| \right) \tag{5.14}$$

for $j = 1{:}p$. This says that

$$\hat{w}_c^{(j)} = \left(R_c^{(j+1)} \cdots R_c^{(p)} \;+\; G^{(j)} \right) \hat{x}_c, \tag{5.15}$$

where $\| G^{(j)} \| \;=\; O(\mathbf{u} \| R_c^{(j+1)} \| \cdots \| R_c^{(p)} \|)$. By rearranging (5.13) and substituting (5.15) we get

$$\begin{aligned} \beta &= (\sigma_1 \cdots \sigma_p (1+\delta_1) - \lambda(1+\delta_2))\, \hat{\gamma} \\ &\quad + \left(\sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1})(1+\mu_j) \tilde{u}_j^T \left(R_c^{(j+1)} \cdots R_c^{(p)} \;+\; G^{(j)} \right) \right) \hat{x}_c \\ &= (\sigma_1 \cdots \sigma_p - \lambda + \epsilon)\, \hat{\gamma} \;+\; \left(\sum_{j=1}^{p} (\sigma_1 \cdots \sigma_{j-1}) u_j^T \left(R_c^{(j+1)} \cdots R_c^{(p)} \right) + e^T \right) \hat{x}_c, \end{aligned}$$

which completely specify $\epsilon \in \mathbb{R}$ and $e \in \mathbb{R}^k$. It follows that (5.10) holds for this choice of ϵ and e. Moreover, (5.11) and (5.12) are both satisfied.

The last thing we must do is verify (5.14) for $j = 1{:}p$. This result is certainly correct if $k = 1$ since $\hat{w}_c = \mathrm{fl}(r_{nn}^{(j+1)} \cdots r_{nn}^{(p)})$. Assume that it holds for general k. Our task is to show that

$$\hat{w}_+^{(j)} = \begin{bmatrix} \hat{\omega}_j \\ \hat{w}_c^{(j)} \end{bmatrix} \;=\; (R_+^{(j+1)} \;+\; F_+^{(j+1)}) \cdots (R_+^{(p)} \;+\; F_+^{(p)}) \hat{x}_+, \tag{5.16}$$

where

$$\| F_+^{(j)} \| \;=\; O\left(\mathbf{u} \| R_+^{(j)} \| \right), \qquad j = p{:}-1{:}1. \tag{5.17}$$

In looking at the specification of the implicit algorithm in section 3, we see that (5.16) holds if $j = p$ since

$$\hat{w}_+^{(p)} = \begin{bmatrix} \hat{\gamma} \\ \hat{x}_c \end{bmatrix} \;=\; \hat{x}_+.$$

Assume that (5.16) and (5.17) hold for some general j that satisfies $1 < j \leq p$. From (5.14) and the definition of ω_j we have

$$\hat{w}_+^{(j-1)} = \begin{bmatrix} \hat{\omega}_{j-1} \\ \hat{w}_c^{(j-1)} \end{bmatrix} = \begin{bmatrix} \mathrm{fl}(\sigma_j \hat{\omega}_j + u_j^T \hat{w}_c^{(j)}) \\ (R_c^{(j)} + F_c^{(j)}) \cdots (R_c^{(p)} + F_c^{(p)}) \hat{x}_c \end{bmatrix}.$$

Since

$$\mathrm{fl}(\sigma_j \hat{\omega}_j + u_j^T \hat{w}_c^{(j)}) = \sigma_j \hat{\omega}_j (1+\tau) + \tilde{u}_j^T \hat{w}_c^{(j)},$$

where $|\tau| = O(\mathbf{u})$, we have

$$\hat{w}_+^{(j-1)} = \begin{bmatrix} \sigma_j(1+\tau) & \tilde{u}_j^T \\ 0 & (R_c^{(j)} + F_c^{(j)}) \end{bmatrix} \begin{bmatrix} \hat{\omega}_j \\ \hat{w}_c^{(j)} \end{bmatrix} = (R_+^{(j)} + F_+^{(j)}) \hat{w}_+^{(j)},$$

where

$$F_+^{(j)} = \begin{bmatrix} \tau\sigma_j & (\tilde{u}_j - u_j)^T \\ 0 & F_c^{(j)} \end{bmatrix}.$$

It follows that (5.16) and (5.17) hold for $j = p\colon -1\colon 1$.

This completes the verification that both the explicit and implicit methods produce computed solutions that satisfy (5.1) and (5.2). We mention that if $\lambda = 0$, then we can solve (1.1) via repeated back-substitution. Using standard results about this process it can be shown that

$$(R^{(1)} + E^{(1)}) \cdots (R^{(p)} + E^{(p)}) \hat{x} = b,$$

where $|E^{(j)}| = O\left(\mathbf{u}|R^{(j)}|\right)$ for $j = 1\colon p$. So although we have shown that the error bounds for the implicit and explicit methods are essentially the same, neither result is as strong as that which can be obtained for the $\lambda = 0$ case.

6. Numerical results. MATLAB implementations of the explicit and implicit methods are available at http://www.cs.cornell.edu/cv/ and were tested to see if the preceding inverse error analysis is realistic. Results like (5.1)–(5.2) that claim a computed solution $\hat{x}$ satisfies a "nearby" system $(A + E)\hat{x} = b$ can be affirmed by comparing the 2-norm of

$$\hat{E} = (b - A\hat{x})\hat{x}^T / (\hat{x}^T \hat{x})$$

with the alleged 2-norm error bound. This is because $(A + \hat{E})\hat{x} = b$ and $\hat{E}$ has the smallest 2-norm of all matrices E that satisfy $(A + E)\hat{x} = b$. Indeed,

$$\| \hat{E} \| = \frac{\| b - A\hat{x} \|}{\| \hat{x} \|}.$$

In our case $A = R^{(1)} \cdots R^{(p)} - \lambda I$, and so the issue before us is the size of

$$\phi(\hat{x}) = \frac{\left(\| b - (R^{(1)} \cdots R^{(p)} - \lambda I)\hat{x} \| / \| \hat{x} \| \right)}{\left(\mathbf{u} \left(\| R^{(1)} \| \cdots \| R^{(p)} \| + |\lambda| \right) \right)} \tag{6.1}$$

TABLE 6.1
Lower and upper bounds for $\| \hat{E}_{imp} \| / \| \hat{E}_{exp} \|$.

	$p = 2$	$p = 4$	$p = 6$
$n = 50$	(.10 , 8.4)	(.08 , 14.6)	(.06 , 7.6)
$n = 100$	(.14 , 6.3)	(.07 , 5.5)	(.06 , 6.6)
$n = 150$	(.09 , 6.0)	(.04 , 5.4)	(.16 , 6.6)
$n = 200$	(.19 , 3.4)	(.09 , 8.7)	(.06 , 8.6)

when $\hat{x}$ is the solution obtained via the implicit and explicit methods. Denote these solutions by $\hat{x}_{imp}$ and $\hat{x}_{exp}$, respectively. Note that if

$$\hat{E}_{imp} = (b - A\hat{x}_{imp})\hat{x}_{imp}^T / (\hat{x}_{imp}^T \hat{x}_{imp}),$$

$$\hat{E}_{exp} = (b - A\hat{x}_{exp})\hat{x}_{exp}^T / (\hat{x}_{exp}^T \hat{x}_{exp}),$$

then from (6.1)

$$\frac{\phi(\hat{x}_{imp})}{\phi(\hat{x}_{exp})} = \frac{\| \hat{E}_{imp} \|}{\| \hat{E}_{exp} \|}.$$

For a particular choice of n and p we experimentally determined a lower bound α and an upper bound β for this quotient, i.e., α and β so that

$$\alpha \| \hat{E}_{exp} \| \leq \| \hat{E}_{imp} \| \leq \beta \| \hat{E}_{exp} \|.$$

In Table 6.1 we report the results. Each cell specifies an estimate of α and β based on 100 randomly generated examples. The results substantiate what the error analysis of section 5 says. The inverse error analysis for the implicit method is essentially the same as the inverse error analysis for the explicit method.

REFERENCES

[1] B. DE MOOR AND P. VAN DOOREN, *Generalizations of the singular value and QR decompositions*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 993–1014.
[2] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[3] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.
[4] C. F. VAN LOAN, *A general matrix eigenvalue algorithm*, SIAM J. Numer. Anal., 12 (1975), pp. 817–834.