SIAM J. Matrix Anal. Appl.
Vol. 25, No. 2, pp. 373–384

FAST AND STABLE ALGORITHMS FOR BANDED PLUS SEMISEPARABLE SYSTEMS OF LINEAR EQUATIONS[*]

S. CHANDRASEKARAN[†] AND M. GU[‡]

Abstract. We present fast and numerically stable algorithms for the solution of linear systems of equations, where the coefficient matrix can be written in the form of a banded plus semiseparable matrix. Such matrices include banded matrices, banded bordered matrices, semiseparable matrices, and block-diagonal plus semiseparable matrices as special cases. Our algorithms are based on novel matrix factorizations developed specifically for matrices with such structures. We also present interesting numerical results with these algorithms.

Key words. banded matrix, bordered matrix, semiseparable matrix, $\mathcal{H}$-matrix, fast algorithms, stable algorithms

AMS subject classifications. 15A09, 15A23, 65F05, 65L10, 65R20

DOI. 10.1137/S0895479899353373

1. Introduction.

In this paper we consider fast and numerically stable solutions of the $n \times n$ linear system of equations

$$A\,x = b, \tag{1.1}$$

where A is the sum of a banded matrix and a semiseparable matrix (see (1.2) below for a definition).

This class of matrices includes banded bordered matrices, which have been discussed in van Huffel and Park [12] and Govaerts [6]. It also includes block-diagonal plus semiseparable matrices, which appear in the numerical solution of boundary-value problems for ordinary differential equations (ODEs) and certain integral equations (see Greengard and Rokhlin [7], Starr [14], and Lee and Greengard [8]). The coefficient matrices generated from domain decomposition methods for partial differential equations (PDEs) tend to have block-diagonal plus bordered structure. Some related work on (1.1) can also be found in Eidelman and Gohberg [3, 4].

1.1. Contributions.

The most important feature of problem (1.1) is that A is a generally dense but highly structured matrix. When A is symmetric positive definite, such a structure can be fully exploited in computing the Cholesky factorization of A. However, the picture changes completely when A is not symmetric positive definite. Although direct methods have been developed for efficient and numerically stable LU and QR factorizations of banded matrices (see Demmel [2, Chap. 2]), such methods do not currently exist for semiseparable matrices, let alone banded plus semiseparable matrices. The main difficulty is that LU and QR factorizations have tremendous difficulties in maintaining both numerical stability and banded plus semiseparable

[*] Received by the editors March 8, 1999; accepted for publication (in revised form) by L. Reichel July 31, 2002; published electronically August 19, 2003.

http://www.siam.org/journals/simax/25-2/35337.html

[†] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560 (shiv@ece.ucsb.edu).

[‡] Department of Mathematics, University of California, Berkeley, CA 94720-3840 (mgu@math.berkeley.edu). The research of this author was supported in part by NSF Career Award CCR-9702866 and by Alfred Sloan Research Fellowship BR-3720.

structure and, consequently, require $O(n^3)$ flops[1] to stably compute such factorizations of A in (1.1).

In this paper, we present a number of fast and numerically stable direct methods for solving (1.1). For banded bordered linear systems of equations, these methods work equally well and are more stable and efficient than those of [12, 6]. Our methods are based on some new matrix factorizations we developed specifically for banded plus semiseparable matrices. We also present interesting results from our numerical experiments with these methods in MATLAB.

1.2. Notation.

To describe the problem precisely, we first introduce some notation. Reminiscent of MATLAB notation, we use $\mathbf{triu}\,(\mathbf{A},\mathbf{k})$ to denote the matrix which is identical to the matrix A on and *above* the kth diagonal. $k=0$ is the main diagonal, $k>0$ is above the main diagonal, and $k<0$ is below the main diagonal. Similarly, $\mathbf{tril}\,(\mathbf{A},\mathbf{k})$ denotes the matrix which is identical to the matrix A on and *below* the kth diagonal. For example,

$$\mathbf{triu}\left(\begin{pmatrix}\alpha & \beta & \gamma\\ \delta & \zeta & \eta\\ \theta & \lambda & \mu\end{pmatrix},\mathbf{1}\right)=\begin{pmatrix}0 & \beta & \gamma\\ 0 & 0 & \eta\\ 0 & 0 & 0\end{pmatrix},\mathbf{tril}\left(\begin{pmatrix}\alpha & \beta & \gamma\\ \delta & \zeta & \eta\\ \theta & \lambda & \mu\end{pmatrix},-\mathbf{1}\right)=\begin{pmatrix}0 & 0 & 0\\ \delta & 0 & 0\\ \theta & \lambda & 0\end{pmatrix}.$$

As a banded plus semiseparable matrix, the matrix A in (1.1) can be written as

$$A = D + \mathbf{triu}\left(\mathbf{u}\,\mathbf{v}^{\mathbf{T}},\mathbf{b_u}+\mathbf{1}\right) + \mathbf{tril}\left(\mathbf{p}\,\mathbf{q}^{\mathbf{T}},-\mathbf{b_l}-\mathbf{1}\right), \tag{1.2}$$

where D is an $n\times n$ banded matrix, with b_u nonzero diagonals strictly above the main diagonal and b_l nonzero diagonals strictly below the main diagonal; u and v are $n\times r_u$ matrices and p and q are $n\times r_l$ matrices.

When $b_u=b_l=0$, D is a diagonal matrix, and A is a diagonal plus semiseparable matrix. When $r_u=r_l=0$, $A=D$ is a banded matrix. We are interested in the numerical solution of the linear system (1.1). The rest of this paper provides a set of numerically backward stable algorithms which take $O\left(n(b_u+b_l+r_u+r_l)^2\right)$ flops to solve (1.1) as opposed to $O(n^3)$ by using traditional methods involving LU and QR factorizations. The exact constant hidden in the $O(\cdot)$ notation varies among our algorithms.

Throughout this paper, we will take the liberty of using I to denote an identity matrix of any dimension.

The rest of this paper is organized as follows. In section 2 we illustrate the basic ideas behind our algorithms through a simple example. In section 3 we describe the algorithms in some detail. In section 4 we present our numerical results with these algorithms.

2. Basic idea.

In this section we give a description of the basic idea in the simple case when D is a diagonal matrix ($b_u=b_l=0$), and u, v, p, and q have only one column ($r_u=r_l=1$).

The idea is to compute a two-sided decomposition of the form

$$A = W\,L\,H\,, \tag{2.1}$$

where W and H can be written as the product of elementary matrices, and L is a lower triangular matrix. The three matrices W, L, H themselves are never explicitly

[1] A *flop* is a floating point operation such as $+, -, \times$, or $\div$.

formed but inverted efficiently online as the algorithm proceeds. In this section, we will choose the matrices W and H to be the products of elementary Givens rotations. When we discuss our algorithms in full detail in section 3, we will allow ourselves the additional freedom of choosing W and H to be products of elementary Householder reflections or Gaussian elimination matrices with column and/or row permutations.

More specifically, consider the 5×5 case,

$$A = \begin{pmatrix} D_0 & u_0v_1 & u_0v_2 & u_0v_3 & u_0v_4 \\ p_1q_0 & D_1 & u_1v_2 & u_1v_3 & u_1v_4 \\ p_2q_0 & p_2q_1 & D_2 & u_2v_3 & u_2v_4 \\ p_3q_0 & p_3q_1 & p_3q_2 & D_3 & u_3v_4 \\ p_4q_0 & p_4q_1 & p_4q_2 & p_4q_3 & D_4 \end{pmatrix}.$$

For future convenience we also assume that the right-hand side is of the form

$$\bar{b} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} - \tau_{-1} \begin{pmatrix} 0 \\ 0 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix},$$

where $\tau_{-1} = 0$. (Of course, the second term on the right-hand side has no effect at this stage, but it will capture the general form of the recursion as we proceed.)

Now suppose that W_0 is a Givens rotation such that

$$W_0 \begin{pmatrix} u_0 \\ u_1 \\ w \end{pmatrix} = \begin{pmatrix} 0 \\ \sqrt{u_0^2 + u_1^2} \\ w \end{pmatrix} \equiv \begin{pmatrix} 0 \\ \widehat{u}_1 \\ w \end{pmatrix} \tag{2.2}$$

for any vector w. Then if we apply W_0 from the left to A, we obtain

$$\widehat{A} = W_0\, A = \begin{pmatrix} \widehat{A}_{00} & \widehat{A}_{01} & 0 & 0 & 0 \\ \widehat{A}_{10} & \widehat{A}_{11} & \widehat{u}_1v_2 & \widehat{u}_1v_3 & \widehat{u}_1v_4 \\ p_2q_0 & p_2q_1 & D_2 & u_2v_3 & u_2v_4 \\ p_3q_0 & p_3q_1 & p_3q_2 & D_3 & u_3v_4 \\ p_4q_0 & p_4q_1 & p_4q_2 & p_4q_3 & D_4 \end{pmatrix}.$$

We also apply W_0 to $\bar{b}$ to obtain

$$\widehat{b} = W_0\, \bar{b} = W_0 \left(\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} - \tau_{-1} \begin{pmatrix} 0 \\ 0 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} \right) = \begin{pmatrix} \widehat{b}_0 \\ \widehat{b}_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} - \tau_{-1} \begin{pmatrix} 0 \\ 0 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix},$$

where we have deliberately written the formula in such a way that it would be correct even if τ_{-1} had not been zero.

We next choose a Givens rotation, H_0, such that

$$H_0^T \begin{pmatrix} \widehat{A}_{00} \\ \widehat{A}_{01} \\ w \end{pmatrix} = \begin{pmatrix} \sqrt{\widehat{A}_{00}^2 + \widehat{A}_{01}^2} \\ 0 \\ w \end{pmatrix} \equiv \begin{pmatrix} \widetilde{A}_{00} \\ 0 \\ w \end{pmatrix} \tag{2.3}$$

for any vector w. Further let

$$H_0^T \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} = \begin{pmatrix} \widetilde{q}_0 \\ \widetilde{q}_1 \end{pmatrix} \quad \text{and} \quad H_0^T \begin{pmatrix} \widehat{A}_{10} \\ \widehat{A}_{11} \end{pmatrix} = \begin{pmatrix} \widetilde{A}_{10} \\ \widetilde{A}_{11} \end{pmatrix}.$$

Then

$$\widetilde{A} = W_0\, A\, H_0 = \widehat{A}\, H_0 = \begin{pmatrix} \widetilde{A}_{00} & 0 & 0 & 0 & 0 \\ \widetilde{A}_{10} & \widetilde{A}_{11} & \widehat{u}_1 v_2 & \widehat{u}_1 v_3 & \widehat{u}_1 v_4 \\ p_2 \widetilde{q}_0 & p_2 \widetilde{q}_1 & D_2 & u_2 v_3 & u_2 v_4 \\ p_3 \widetilde{q}_0 & p_3 \widetilde{q}_1 & p_3 q_2 & D_3 & u_3 v_4 \\ p_4 \widetilde{q}_0 & p_4 \widetilde{q}_1 & p_4 q_2 & p_4 q_3 & D_4 \end{pmatrix}.$$

Now let

$$H_0^{-1}\, x = H_0^{-1} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \equiv \begin{pmatrix} \widetilde{\chi}_0 \\ \widetilde{x}_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \equiv \widetilde{x}. \tag{2.4}$$

Then it follows from $W_0\, A\, H_0\, H_0^{-1}\, x = \widetilde{A}\, \widetilde{x} = W_0\, b = \widehat{b}$ that

$$\widetilde{\chi}_0 = \frac{\widehat{b}_0}{\widetilde{A}_{00}}.$$

Also let

$$\tau_0 = \tau_{-1} + \widetilde{\chi}_0\, \widetilde{q}_0, \quad \widetilde{b}_1 = \widehat{b}_1 - \widetilde{\chi}_0\, \widetilde{A}_{10}, \quad \text{and} \quad \widetilde{b}_2 = b_2 - \tau_0\, p_2. \tag{2.5}$$

To reach this stage we needed to compute all the "tilde" and "hat" quantities except $\widetilde{x}_1$. They can be computed in *constant* time, independent of the size of the matrix A.

Now we can proceed to solve the smaller 4×4 system of equations,

$$\begin{pmatrix} \widetilde{A}_{11} & \widehat{u}_1 v_2 & \widehat{u}_1 v_3 & \widehat{u}_1 v_4 \\ p_2 \widetilde{q}_1 & D_2 & u_2 v_3 & u_2 v_4 \\ p_3 \widetilde{q}_1 & p_3 q_2 & D_3 & u_3 v_4 \\ p_4 \widetilde{q}_1 & p_4 q_2 & p_4 q_3 & D_4 \end{pmatrix} \begin{pmatrix} \widetilde{x}_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} \widetilde{b}_1 \\ \widetilde{b}_2 \\ b_3 \\ b_4 \end{pmatrix} - \tau_0 \begin{pmatrix} 0 \\ 0 \\ p_3 \\ p_4 \end{pmatrix},$$

which is exactly like the original 5×5 system of equations in form. That is, the coefficient matrix is a diagonal matrix plus a semiseparable matrix, and the right-hand side is also of the requisite form. Hence we can use this recursion ((2.2) through (2.5)) three times until the problem size becomes two, at which point we solve the system directly. Let the five numbers obtained by this recursion, $\widetilde{\chi}_0$, $\widetilde{\chi}_1$, $\widetilde{\chi}_2$, $\widetilde{\chi}_3$, and $\widetilde{\chi}_4$, be the components of the five-dimensional vector χ. Then it follows from (2.4) that the actual solution x to the original 5×5 system of equations is given by

$$x = H_0\, H_1\, H_2\, \chi, \tag{2.6}$$

where the H_i's are the successive Givens transforms computed from the recursion (2.3) but set up in such a way that they only affect rows i and $i + 1$. Since there are only three of these transforms, we retain the linear time complexity of the algorithm.

The backward stability of the algorithm follows from the fact that we use only orthogonal transforms and a single forward substitution.

Our factorization is similar in form to the ULV factorization proposed by Stewart [15]. However, the ULV factorization of Stewart is developed primarily to reveal potential numerical rank-deficiency in a general matrix and can take $O(n^3)$ flops to compute, whereas our factorization is designed primarily to take advantage of the banded plus semiseparable structure for large savings in computational cost without sacrificing numerical stability.

There are two places in the recursion where elimination is necessary. In (2.2) we chose W_0 to be a Givens rotation to eliminate u_0, and in (2.3) we chose P_0 to be another Givens rotation to eliminate $\widehat{A}_{01}$. These transformations can be replaced by Householder transformations or Gaussian elimination matrices with row or column pivoting for general banded plus semiseparable matrices. This results in several algorithms with different efficiency and numerical stability properties. In the next section, we describe a general procedure for solving (1.1) via the computation of the factorization (2.1). We also discuss efficiency and numerical stability issues for different choices of W and H in (2.1).

3. The algorithms.

We now describe fast algorithms for solving (1.1), where A is a general banded plus semiseparable matrix of the form (1.2).

3.1. Preprocessing and basic linear algebra procedures.

Some preprocessing is needed before the algorithms formally start. We make u lower triangular by computing a QR factorization $u^T = Q\,R$ and resetting

$$u := R^T \quad \text{and} \quad v := v\,Q. \tag{3.1}$$

This operation takes roughly $6nr_u^2$ flops using the fact that Q is computed in factored form [5, Chap. 5].

We also review a few well-known basic linear algebra routines needed in our algorithms. Let L be an $m \times s$ lower triangular matrix with $m > s$,

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{s1} & l_{s2} & \cdots & l_{ss} \\ \vdots & \vdots & & \vdots \\ l_{m1} & l_{m2} & \cdots & l_{ms} \end{pmatrix}.$$

Algorithm 3.1 below is a standard procedure for efficiently zeroing out entries $l_{11}, l_{22}, \ldots, l_{ss}$ on the main diagonal of L by using s Givens rotations (see [5, Chap. 12]).

Algorithm 3.1. *Elimination with Givens rotations.*

```
for $i := s$ to $1$ step $-1$ do
        • Choose $c_i^2 + s_i^2 = 1$ such that
              $\begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix} \begin{pmatrix} l_{i,i} \\ l_{i+1,i} \end{pmatrix} = \begin{pmatrix} 0 \\ \rho_i \end{pmatrix}$,  where  $\rho_i = \sqrt{l_{i,i}^2 + l_{i+1,i}^2}$.
        • Set $l_{i,i} := 0$, $l_{i+1,i} := \rho_i$, and compute
              $\begin{pmatrix} l_{i,1} & \cdots & l_{i,i-1} \\ l_{i+1,1} & \cdots & l_{i+1,i-1} \end{pmatrix} := \begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix} \begin{pmatrix} l_{i,1} & \cdots & l_{i,i-1} \\ l_{i+1,1} & \cdots & l_{i+1,i-1} \end{pmatrix}$.
endfor
```

Let W be the product of all the Givens rotations used in the above algorithm. Then its output can be written via a matrix-matrix product as $L := W\,L$.

Similarly, we can zero out the main diagonal of L by using a banded Gaussian elimination procedure with row pivoting. See Golub and Van Loan [5, Chap. 4] for details.

Let $G \in \mathbf{R}^{m\times s}$ be a general dense matrix. Then we can choose a Householder transformation

$$H = I - 2u\,u^T \quad \text{with} \quad \|u\|_2 = 1$$

to zero out all the entries in the first row of G except the $(1,1)$ entry as follows:

$$G\,H = \begin{pmatrix} \widehat{\gamma} & 0 \\ \widehat{g} & \widehat{G} \end{pmatrix}. \tag{3.2}$$

The cost for computing u is $O(s)$, and the cost for computing $G\,H$ is about $4ms$ flops (see [5, Chap. 5]).

Alternatively, we can choose H in (3.2) to be a Gaussian elimination matrix of the form

$$H = \begin{pmatrix} 1 & -h^T \\ 0 & I \end{pmatrix}.$$

Column pivoting can be used to enhance numerical stability. The cost for computing $G\,H$ is about $2ms$ flops (see Golub and Van Loan [5, Chap. 3]).

3.2. New algorithms. Let $\ell = b_u + r_u + 1$ and $m = \ell + b_l$; we begin by writing A in the form

$$A = \begin{pmatrix} G & E \\ C & F \end{pmatrix}, \tag{3.3}$$

where $G \in \mathbf{R}^{m\times\ell}$ is a dense matrix (its banded plus semiseparable structure will be ignored); $F \in \mathbf{R}^{(n-r_u-1)\times(n-\ell)}$ is a banded plus semiseparable rectangular matrix; and both $C \in \mathbf{R}^{(n-m)\times\ell}$ and $E \in \mathbf{R}^{(r_u+1)\times(n-\ell)}$ are low rank matrices. We caution that, strictly speaking, (3.3) is *not* a block partitioning of A, since the row dimension of G is larger than that of E in general.

In further detail, we write $C = \bar{p}\,\bar{q}^T$, where $\bar{p} \in \mathbf{R}^{(n-m)\times r_l}$ and $\bar{q} \in \mathbf{R}^{\ell\times r_l}$ contain the last $n-m$ rows of p and the first ℓ rows of q, respectively. Similarly, $E = \bar{u}\,\bar{v}^T$, where $\bar{u} \in \mathbf{R}^{(r_u+1)\times r_u}$ and $\bar{v} \in \mathbf{R}^{(n-\ell)\times r_u}$ contain the first $r_u + 1$ rows of u and the last $n - \ell$ rows of v, respectively. As suggested in section 3.1, we assume that $\bar{u}$ is a lower triangular matrix.

As in section 2, we will solve (1.1) by recursively solving the linear system

$$A\,x = \bar{b} \equiv b - \begin{pmatrix} 0 \\ \bar{p}\,\tau \end{pmatrix}, \tag{3.4}$$

where $\tau_{-1} = 0 \in \mathbf{R}^{r_l}$ is an auxiliary vector that will play the role of scalar τ_{-1} in the example in section 2. As before, we will compute a two-sided decomposition (2.1) of A and invert matrices W, L, and H on the fly.

To start the recursion, we use Algorithm 3.1 to compute a matrix W_0 so that $W_0\,\bar{u}$ is a lower triangular matrix with zeros on its main diagonal. Compute

$$\begin{pmatrix} 0 \\ \widehat{u} \end{pmatrix} := W_0\,\bar{u}, \quad \widehat{G} := \begin{pmatrix} W_0 & 0 \\ 0 & I \end{pmatrix} G, \quad \text{and} \quad \widehat{b} := \begin{pmatrix} W_0 & 0 \\ 0 & I \end{pmatrix} b - \begin{pmatrix} 0 \\ \bar{p}\,\tau_{-1} \end{pmatrix}. \tag{3.5}$$

Linear system (3.4) now becomes

$$\begin{pmatrix} \widehat{G} & \begin{pmatrix} 0 \\ \widehat{u} \end{pmatrix} \bar{v}^T \\ \bar{p}\,\bar{q}^T & F \end{pmatrix} x = \widehat{b}.$$

We further choose H_0 to zero out the first row of $\widehat{G}$ except the $(1,1)$ entry. Compute

$$\begin{pmatrix} \widetilde{\gamma} & 0 \\ \widetilde{g} & \widetilde{G} \end{pmatrix} := \widehat{G}\,H_0 \quad \text{and} \quad \begin{pmatrix} \widetilde{\rho}^T \\ \widetilde{q} \end{pmatrix} := H_0^T\,\bar{q}. \tag{3.6}$$

Linear system (3.4) now has the following form:

$$\begin{pmatrix} \widetilde{\gamma} & 0 & 0 \\ \widetilde{g} & \widetilde{G} & \widehat{u}\,\bar{v}^T \\ \bar{p}\,\widetilde{\rho} & \bar{p}\,\widetilde{q}^T & F \end{pmatrix} \widetilde{x} = \widehat{b}, \tag{3.7}$$

where

$$\widetilde{x} = \begin{pmatrix} H_0^{-1} & 0 \\ 0 & I \end{pmatrix}, \quad x = \begin{pmatrix} \widetilde{\chi}_0 \\ \widetilde{x}_1 \\ \widetilde{x}_2 \end{pmatrix}, \quad \text{and} \quad \widehat{b} = \begin{pmatrix} \widehat{\beta} \\ \widehat{b}_1 \\ \widehat{b}_2 - \bar{p}\,\tau_{-1} \end{pmatrix}.$$

Now we can perform one step of forward substitution in (3.7) to get $\widetilde{\chi}_0 = \widehat{\beta}/\widetilde{\gamma}$ and

$$\begin{pmatrix} \widetilde{G} & \widehat{u}\,\bar{v}^T \\ \bar{p}\,\widetilde{q}^T & F \end{pmatrix} \widetilde{x} = \begin{pmatrix} \widehat{b}_1 - \widetilde{\chi}_0\,\widetilde{g} \\ \widehat{b}_2 - \bar{p}\,\left(\tau_{-1} + \widetilde{\chi}_0\,\widetilde{\rho}\right) \end{pmatrix} \equiv \begin{pmatrix} \widetilde{b}_1 \\ \widehat{b}_2 - \bar{p}\,\tau_0 \end{pmatrix}. \tag{3.8}$$

This is a system smaller in dimension than (3.4). To complete the recursion, in the following we rewrite it in the form of (3.4):

$$\bar{v} = \begin{pmatrix} \widetilde{\nu}^T \\ \widetilde{v} \end{pmatrix}, \quad \bar{p} = \begin{pmatrix} \widetilde{\pi}^T \\ \widetilde{p} \end{pmatrix}, \quad \text{and} \quad F = \begin{pmatrix} \widetilde{f}_1 & \widetilde{f}_2^T \\ \widetilde{f}_3 & \widetilde{F} \end{pmatrix},$$

where $\widetilde{\nu}^T$ and $\widetilde{\pi}^T$ are the first rows of $\bar{v}$ and $\bar{p}$, respectively; $\widetilde{f}_1 \in \mathbf{R}^{m-r_u}$; and $\widetilde{f}_2$ and $\widetilde{f}_3$ are column vectors of appropriate dimensions. Similarly to (3.3), the block form of F above is, strictly speaking, *not* a block partitioning of F, since the length of $\widetilde{f}_1$ is larger than 1 in general. $\widetilde{F}$ is itself a banded plus semiseparable rectangular matrix.

With this notation, we can now rewrite (3.8) in the form of (3.4) as

$$\begin{pmatrix} \dot{G} & \dot{E} \\ \dot{C} & \tilde{F} \end{pmatrix} \widetilde{x} = \dot{b} - \begin{pmatrix} 0 \\ \widetilde{p}\,\tau_0 \end{pmatrix}, \tag{3.9}$$

where

$$\dot{G} = \begin{pmatrix} \widetilde{G} & \widehat{u}\,\widetilde{\nu} \\ \widetilde{\pi}^T\,\widetilde{q}^T & \widetilde{f}_1 \end{pmatrix}, \quad \dot{C} = \widetilde{p}\,\begin{pmatrix} \widetilde{q}^T & \widetilde{\phi} \end{pmatrix}, \quad \dot{E} = \begin{pmatrix} \widehat{u} \\ \widehat{\mu}^T \end{pmatrix} \widetilde{v}^T, \quad \dot{b} = \begin{pmatrix} \widetilde{b}_1 \\ \widehat{b}_2 - \begin{pmatrix} \widetilde{\pi}^T\,\tau_0 \\ 0 \end{pmatrix} \end{pmatrix},$$

with $\widetilde{\phi}^T$ and $\widehat{\mu}^T$ being the $(\ell+1)$th and (r_u+2)th rows of q and u, respectively. Once again the block form of $\dot{G}$ is not a block partitioning.

As in section 2, we can perform elimination and forward substitution steps using formulas (3.4) through (3.9) recursively for some k times to obtain solution components $\widetilde{\chi}_0, \widetilde{\chi}_1, \ldots, \widetilde{\chi}_{k-1}$. We stop the recursion when the problem size $n-k$ in (3.9) becomes so small that $n-k \approx m$, at which point we solve it directly to get a solution $\widetilde{\chi}$.

To recover the solution to our original problem (1.1), let $H_0, H_1, \ldots, H_{k-1}$ be the elimination matrices used at the second elimination step defined by (3.6) and (3.7). We compute the solution to (1.1) as

$$x = \begin{pmatrix} I & 0 & 0 \\ 0 & H_0 & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & H_1 & 0 \\ 0 & 0 & I \end{pmatrix} \cdots \begin{pmatrix} I & 0 & 0 \\ 0 & H_{k-1} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \widetilde{\chi}_0 \\ \vdots \\ \widetilde{\chi}_{k-1} \\ \widetilde{\chi} \end{pmatrix}, \tag{3.10}$$

where the various identity matrices I are, in general, of different dimensions.

3.3. Efficiency and numerical stability considerations.

In this section we consider special choices of matrices W and H in the recursion and how they affect the efficiency and numerical stability of the procedure. To make flop counting simpler, in this section we assume that $1 \ll r_l, r_u, b_l, b_u \ll n$ even though our algorithms work for general banded plus semiseparable matrices.

For complete backward stability, we can choose the W_0 matrices in (3.5) to be the product of r_u Givens rotations as suggested by Algorithm 3.1. The costs for computing $\widehat{u}$, $\widehat{G}$, and $\widehat{b}$ are about $3r_u^2$ flops, $6r_u\ell$ flops, and $6r_u$ flops, respectively. Hence the total cost for one step of (3.5) is about $3r_u(r_u+2\ell)$ flops.

We then choose H_0 in (3.6) as a Householder transformation. The costs for computing $\widehat{G}\,H_0$ and $H_0^T\,\bar{q}$ are about $4m\ell$ flops and $4r_l\ell$ flops, respectively. Hence the total cost for one step of (3.6) is about $4(m+r_l)\ell$ flops.

In (3.8), the costs for computing $\widetilde{b}_1$ and τ_0 are about $2m$ flops and $2r_l$ flops, respectively, leading to a total of $2(m+r_l)$ flops.

In (3.9), the main cost is to explicitly form the last row and column of $\dot{G}$. The costs for computing $\widehat{u}\,\widetilde{\nu}$ and $\widetilde{\pi}^T\,\widetilde{q}^T$ are about $2r_u^2$ flops and $2r_l\ell$ flops, respectively. There is essentially no cost for $\widetilde{f}_1$, which consists of the nonzero components of a column in the banded matrix D. Hence the total cost in (3.9) is about $2(r_u^2 + r_l\ell)$ flops.

Since there are $k \approx n$ steps of recursion, the total cost for the procedure is about

$$\begin{aligned} &\big(3r_u(r_u+2\ell)+4(m+r_l)\ell+2(r_u^2+r_l\ell)\big)\,n \\ &\quad = \big(5r_u^2+2\,(2b_u+2b_l+3r_l+5r_u)\,(b_u+r_u)\big)\,n \quad \text{flops.} \end{aligned} \tag{3.11}$$

Additionally, there is a cost of about $6r_u^2 n$ flops for the preprocessing step (3.1).

With such choices of W_0 and H_0, we obtain a factorization (2.1) with orthogonal matrices W and H. Since only orthogonal transformations and one forward substitution are used for the solution of (1.1), this algorithm is backward stable.

To reduce computational cost, we can also choose W_0 via the banded Gaussian elimination procedure with row pivoting in Golub and Van Loan [5, Chap. 4]. Also, we can choose H_0 as a Gaussian elimination matrix with column pivoting. This choice of W_0 and H_0 leads to a factorization (2.1) with upper triangular matrices W and H. It is quite interesting to note that factorizations of this form do not seem to have been previously discussed in the literature.

With this choice of W_0 and H_0, the cost for one step of (3.5) is about $r_u(r_u+2\ell)$ flops; the cost for one step of (3.6) is about $2(m+r_l)\ell$ flops; and the total cost in (3.9) is about $2(r_u^2+r_l\ell)$ flops. With $k \approx n$ steps of recursion, the total cost for the procedure is about

$$\begin{aligned} &\big(r_u(r_u+2\ell)+2(m+r_l)\ell+2(r_u^2+r_l\ell)\big)\,n \\ &\quad = \big(3r_u^2+2\,(b_u+2b_l+2r_l+2r_u)\,(b_u+r_u)\big)\,n \quad \text{flops.} \end{aligned} \tag{3.12}$$

Additionally, there is a cost of about $6r_u^2 n$ flops for the preprocessing step (3.1).

It is well known that Gaussian elimination with partial pivoting could occasionally become numerically unstable if certain element growth is too large (see Golub and Van Loan [5, Chap. 3]). Thus, the numerical stability of Gaussian elimination procedures in (3.5) and (3.6) could not be guaranteed for a large value of r_u or b_u. In fact, the above procedure will be unstable for the case where $r_l = r_u = 0$, $b_l = b_u = k \gg 1$, and the first k rows of A are all zero, except leading k columns which contain the $k \times k$ matrix A_1 where (see Golub and Van Loan [5, Chap. 3])

$$A_1' = \begin{pmatrix} 1 & & & & 1 \\ -1 & 1 & & & 1 \\ \vdots & \ddots & \ddots & & \vdots \\ -1 & \cdots & -1 & 1 & 1 \\ -1 & \cdots & -1 & -1 & 1 \end{pmatrix}.$$

Alternatively, we can choose only one of W_0 and H_0 to be orthogonal, leading to a factorization (2.1) with one of W and H orthogonal and the other upper triangular. Furthermore, the choices of W_0 and H_0 can change from one recursion step to another, leading to a factorization (2.1) with no obvious structures in W and H.

While our algorithms were presented in such a way that only one variable in (3.4) is eliminated in forward substitution at every recursion step, it is possible to reorganize the computation to develop a block version where a number of variables are all eliminated at the same time. Given the success of blocking in the recent linear algebra package LAPACK [1], it seems clear that when the dimension becomes very large, the problem (1.1) can be solved more efficiently by block versions of our algorithms.

Finally, we note that the problem (1.1) can be rewritten in the form

$$B\,y = S\,b, \quad B = S\,A\,S, \quad \text{and} \quad x = S\,y, \tag{3.13}$$

where S is the matrix with ones on the main antidiagonal and zero elsewhere.[2] It is easy to verify that

$$B = (S\,D\,S) + \mathbf{tril}\left((\mathbf{S\,u})\ (\mathbf{S\,v})^{\mathbf{T}}, -\mathbf{b_u} - \mathbf{1}\right) + \mathbf{triu}\left((\mathbf{S\,p})\ (\mathbf{S\,q})^{\mathbf{T}}, \mathbf{b_l} + \mathbf{1}\right).$$

It can be verified that $(S\,D\,S)$ is a banded matrix with b_u nonzero diagonals strictly below the main diagonal and b_l nonzero diagonals strictly above the main diagonal. Hence, B is itself a banded plus semiseparable matrix with the banded plus semiseparable structure of A'. Applying the two algorithms we just discussed to solve (3.13), we see that the total costs are

$$\left(5r_l^2 + 2\left(2b_l + 2b_u + 3r_u + 5r_l\right)\left(b_l + r_l\right)\right) n \quad \text{flops} \tag{3.14}$$

and

$$\left(3r_l^2 + 2\left(b_l + 2b_u + 2r_u + 2r_l\right)\left(b_l + r_l\right)\right) n \quad \text{flops,} \tag{3.15}$$

respectively. This suggests that one should choose among the two forms (1.1) and (3.13) according to formulas (3.11), (3.12), (3.14), and (3.15) to reduce computational cost.

4. Numerical experiments.

In this section, we summarize the results from our numerical experiments with the algorithms that were presented in section 3. These experiments were performed on an UltraSparc 2 workstation in MATLAB with double precision $\epsilon \approx 2 \times 10^{-16}$.

We tested the following two algorithms:

- Algorithm I: Only Gaussian elimination steps with partial pivoting were used in computing (2.1).
- Algorithm II: Only Givens rotations and Householder reflections were used in computing (2.1).

In all of the test matrices, we chose $r_l = n/10$, $r_u = n/250$, $b_u = 10$, and $b_l = 10$. The matrix entries were generated randomly.

In Table 4.1, we compared Algorithms I and II in terms of the numbers of flops required to solve (1.1). The column marked GEPP is the number of flops required for Gaussian elimination with partial pivoting to solve (1.1) by treating A as a dense matrix. We see that Algorithm I requires less flops than Algorithm II, and both Algorithms I and II require significantly fewer flops than GEPP to solve (1.1).

In Table 4.2, we compared Algorithms I and II in terms of execution times and backward errors. The execution times are in seconds, and the backward error is defined as

$$\frac{\|A\,\widehat{x} - b\|_\infty}{\|A\|_\infty\,\|\widehat{x}\|_\infty},$$

where $\widehat{x}$ is the computed solution to (1.1). This backward error is the smallest relative backward error in the ∞-norm (see [5, Chap. 3]). Clearly Algorithm I is faster than Algorithm II as expected. Both are comparable in terms of backward errors. However, as we mentioned in section 3, the numerical stability of Algorithm I could not be guaranteed for a large value of b_u or r_u.

[2] For example, when $n = 2$, we have

$$S = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Table 4.1
Numbers of flops.

n	Algorithm I	Algorithm II	GEPP
250	5.5×10^{5}	8.7×10^{5}	1.0×10^{7}
500	2.1×10^{6}	3.2×10^{6}	8.3×10^{7}
750	4.7×10^{6}	7.2×10^{6}	2.8×10^{8}
1000	8.7×10^{6}	1.3×10^{7}	6.7×10^{8}
1250	1.4×10^{7}	2.2×10^{7}	1.3×10^{9}
1500	2.2×10^{7}	3.3×10^{7}	2.3×10^{9}
1750	3.1×10^{7}	4.7×10^{7}	3.6×10^{9}
2000	4.3×10^{7}	6.4×10^{7}	5.3×10^{9}
2250	5.6×10^{7}	8.4×10^{7}	7.6×10^{9}
2500	7.3×10^{7}	1.1×10^{8}	1.0×10^{10}

Table 4.2
Execution times and backward errors.

	Time (seconds)		Backward Error	
n	Algorithm I	Algorithm II	Algorithm I	Algorithm II
250	7.6×10^{-1}	1.0×10^{0}	6.1×10^{-19}	1.6×10^{-18}
500	2.2×10^{0}	3.2×10^{0}	1.5×10^{-19}	5.8×10^{-19}
750	4.5×10^{0}	5.7×10^{0}	3.6×10^{-20}	2.0×10^{-19}
1000	8.4×10^{0}	1.1×10^{1}	6.1×10^{-20}	2.0×10^{-19}
1250	1.3×10^{1}	1.6×10^{1}	4.8×10^{-20}	6.3×10^{-20}
1500	1.9×10^{1}	2.3×10^{1}	5.4×10^{-20}	3.8×10^{-19}
1750	2.5×10^{1}	3.1×10^{1}	2.8×10^{-20}	2.9×10^{-20}
2000	3.3×10^{1}	4.1×10^{1}	4.3×10^{-20}	5.3×10^{-20}
2250	4.1×10^{1}	5.1×10^{1}	5.0×10^{-20}	3.4×10^{-19}
2500	5.2×10^{1}	6.3×10^{1}	5.5×10^{-20}	2.2×10^{-19}

5. Conclusions and future work. In this paper we presented fast and numerically stable algorithms for the solution of linear systems of equations, where the coefficient matrix has the banded plus semiseparable structure (1.2). We also presented numerical results that clearly showed the stability and efficiency of these methods. It turns out that the two-sided elimination approach developed in this paper can be applied to a much broader class of matrices, including the $\mathcal{H}$-matrices of Hackbusch and his colleagues [9, 10, 11]. Our future work will concentrate on generalizing our methods to efficiently and stably solve linear systems of equations involving these and other structured matrices.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, 1999.

[2] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.

[3] Y. Eidelman and I. Gohberg, *Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices*, Comput. Math. Appl., 33 (1997), pp. 69–79.

[4] Y. Eidelman and I. Gohberg, *A look-ahead block Schur algorithm for diagonal plus semiseparable matrices*, Comput. Math. Appl., 35 (1998), pp. 25–34.

[5] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[6] W. Govaerts, *Stable solvers and block elimination for bordered systems*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 469–483.

[7] L. Greengard and V. Rokhlin, *On the numerical solution of two-point boundary value problems*, Comm. Pure Appl. Math., 44 (1991), pp. 419–452.

[8] J.-Y. Lee and L. Greengard, *A fast adaptive numerical method for stiff two-point boundary value problems*, SIAM J. Sci. Comput., 18 (1997), pp. 403–429.

[9] W. Hackbusch and B. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic: General complexity estimates*, J. Comput. Appl. Math., 125 (2000), pp. 479–501.

[10] W. Hackbusch, B. N. Khoromskij, and S. A. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, H.-J. Bungartz, R. H. W. Hoppe, and C. Zenger, eds., Springer-Verlag, Berlin, 2000, pp. 9–29.

[11] W. Hackbusch and B. N. Khoromskij, *Towards $\mathcal{H}$-matrix approximation of linear complexity*, in Problems and Methods in Mathematical Physics, J. Elschner, I. Hohberg, and B. Silbermann, eds., Birkhäuser, Basel, 2001, pp. 194–220.

[12] S. van Huffel and H. Park, *Efficient reduction algorithms for bordered band matrices*, J. Numer. Linear Algebra Appl., 2 (1995), pp. 95–113.

[13] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[14] H. P. Starr, Jr., *On the Numerical Solution of One-Dimensional Integral and Differential Equations*, Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, 1992.

[15] G. W. Stewart, *Updating a rank-revealing ULV decomposition*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 494–499.