BLOCK ALGORITHMS FOR ORTHOGONAL SYMPLECTIC FACTORIZATIONS

D. KRESSNER[1] *

[1] *Institut für Mathematik MA 4-5, TU Berlin, D-10623 Berlin, FRG. email: kressner@math.tu-berlin.de*

Abstract.

On the basis of a new WY-like representation block algorithms for orthogonal symplectic matrix factorizations are presented. Special emphasis is placed on symplectic QR and URV factorizations. The block variants mainly use level 3 (matrix-matrix) operations that permit data reuse in the higher levels of a memory hierarchy. Timing results show that our new algorithms outperform standard algorithms by a factor 3–4 for sufficiently large problems.

AMS subject classification: 65F25, 15A23, 65P10, 65F15

Key words: Block algorithms, orthogonal symplectic factorizations, level 3 BLAS

1 Introduction

There are two distinct classes of matrix factorizations considered in this paper, one-sided and two-sided orthogonal factorizations. Decomposing a matrix into a product of an orthogonal matrix and a triangular matrix, e.g. by QR factorization, belongs to the first class as the orthogonal factor only appears on one side of the product. Preprocessing steps for eigenvalue computations, e.g., bidiagonalization and Hessenberg reduction, typically consist of factorizations into products of three matrices involving orthogonal matrices on both sides of the product and thus belong to the second class.

Many of these matrix factorizations have found their way into the LAPACK software library [2]. In these implementations, efficiency is attained by employing WY representations of the involved orthogonal transformations [7, 9, 10, 16]. The application of such representations can be formulated in terms of matrix-matrix multiplications leading to reduced memory traffic which in turn means better performance. For example, on an average work station, computing the Hessenberg form of a 1000×1000 matrix would take more than thrice the time if no WY representations were used.

On the other hand, if the matrix to be factorized or the factors themselves are structured then accuracy considerations require the factorization algorithm to respect the underlying structures. Often, structure exploitation also reduces the number of operations necessary to compute the factorization. For these reasons,

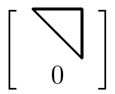
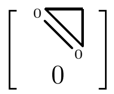
*Supported through the DFG Research Center **Mathematics for Key Technologies**.

it is widely appreciated that structure gives the potential to develop more efficient and more accurate algorithms. In practice, however, there is usually a long way to go in order to develop such an algorithm with an implementation so that the latter is competitive with the corresponding unstructured implementation in LAPACK. In this paper we will discuss this issue for orthogonal symplectic factorizations.

Definition 1.1. *Let* $J := \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}$*, where* I_n *is the* $n \times n$ *identity matrix. A matrix* $\mathcal{Q} \in \mathbb{R}^{2n \times 2n}$ *is* symplectic *if and only if* $\mathcal{Q}J\mathcal{Q}^T = \mathcal{Q}^T J \mathcal{Q} = J$ *and* orthogonal symplectic *if and only if additionally* $\mathcal{Q}^T\mathcal{Q} = \mathcal{Q}\mathcal{Q}^T = I_{2n}$ *holds.*

The following one-sided orthogonal symplectic factorization is used in methods for the symplectic integration of Hamiltonian systems [14] and for the computation of orthogonal bases of Lagrangian subspaces [3, 11].

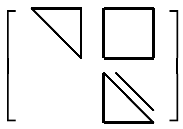
Lemma 1.1. *[8] Let* $\mathcal{A} \in \mathbb{R}^{2m \times n}$ *with* $m \geq n$*, then there exists an orthogonal symplectic matrix* $\mathcal{Q}$ *so that* $\mathcal{A} = \mathcal{Q}\mathcal{R}$ *and*

(1.1) $\mathcal{R} = \begin{bmatrix} R_{11} \\ R_{21} \end{bmatrix}, \quad R_{11} =$  $, \quad R_{21} =$  ,

that is, the matrix $R_{11} \in \mathbb{R}^{m \times n}$ *is upper triangular and* $R_{21} \in \mathbb{R}^{m \times n}$ *is strictly upper triangular.*

Benner, Mehrmann and Xu [5, 6] developed methods for computing eigenvalues and certain invariant subspaces of Hamiltonian matrices. The algorithms presented therein are based on the so called symplectic URV factorization, a two-sided orthogonal symplectic factorization.

Lemma 1.2. *[6] Let* $\mathcal{A} \in \mathbb{R}^{2n \times 2n}$*, then there exist orthogonal symplectic matrices* $\mathcal{U}$ *and* $\mathcal{V}$ *so that* $\mathcal{A} = \mathcal{U}\mathcal{R}\mathcal{V}^T$ *and*

(1.2) $\mathcal{R} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} =$  ,

that is, the matrix $R_{21} \in \mathbb{R}^{n \times n}$ *is zero,* $R_{11} \in \mathbb{R}^{n \times n}$ *is upper triangular and* $R_{22} \in \mathbb{R}^{n \times n}$ *is lower Hessenberg.*

The algorithm presented in [6] for computing the symplectic URV factorization requires roughly the same number of operations as the standard algorithm [12, p.344] for computing the Hessenberg form does. However, for the reasons described above, the latter algorithm can be implemented very efficiently while the necessary tools, WY-like representations, are not available for orthogonal symplectic factorizations.

In Section 2 we present such a WY-like representation which despite its more complicated nature shares the same favorable properties as standard WY representations. In Sections 3 and 4 we apply this representation to develop block algorithms for the symplectic QR and the symplectic URV factorization, respectively. It is shown in Section 5 that numerical stability of these algorithms is preserved. Section 6 is devoted to numerical results showing that block algorithms lead to more efficient implementations.

2 A WY-like representation for products of elementary orthogonal symplectic matrices

An orthogonal matrix $\mathcal{Q} \in \mathbb{R}^{2n\times 2n}$ is symplectic if and only if it takes the form $\mathcal{Q} = \left[\begin{smallmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{smallmatrix}\right]$, where $Q_1, Q_2 \in \mathbb{R}^{n\times n}$ [15]. Furthermore, orthogonal symplectic matrices can be decomposed into products of the following two types of elementary matrices. These are $2n \times 2n$ Givens rotation matrices of the form

$$G_j(\theta) = \left[\begin{array}{ccccc} I_{j-1} & & & & \\ & \cos\theta & & \sin\theta & \\ & & I_{n-1} & & \\ & -\sin\theta & & \cos\theta & \\ & & & & I_{n-j} \end{array}\right], \quad 1 \le j \le n,\ \theta \in [-\pi/2, \pi/2),$$

and the direct sum of two identical $n \times n$ Householder matrices

$$H_j(v,\beta) = \left[\begin{array}{cc} I_n - \beta vv^T & \\ & I_n - \beta vv^T \end{array}\right],$$

where v is a vector of length n with its first $j-1$ elements equal to zero. Clearly, both matrices are orthogonal and symplectic. A simple combination of these transformations can be used to map an arbitrary vector $x \in \mathbb{R}^{2n}$ into the linear space

$$\mathcal{E}_j = \operatorname{span}\{e_1, \ldots, e_j, e_{n+1}, \ldots, e_{n+j-1}\},$$

where e_i denotes the i-th unit vector of length $2n$. Note that in the following algorithm elements $1, \ldots, j-1$ and $n+1, \ldots, n+j-1$ of the vector x are unaffected.

Algorithm 2.1.
Input: *A vector $x \in \mathbb{R}^{2n}$ and an index $j \le n$.*
Output: *Vectors $v, w \in \mathbb{R}^n$ and $\beta, \gamma, \theta \in \mathbb{R}$ so that*

$$(H_j(v,\beta)G_j(\theta)H_j(w,\gamma))^T x \in \mathcal{E}_j.$$

1. Determine $v \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ such that the last $n-j$ elements of $x \leftarrow H_j(v,\beta)x$ are zero [12, p.209].
2. Determine $\theta \in [-\pi/2, \pi/2)$ such that the $(n+j)$-th element of $x \leftarrow G_j(\theta)x$ is zero [12, p.215].
3. Determine $w \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$ such that the $(j+1)$-th to the n-th elements of $x \leftarrow H_j(w,\gamma)x$ are zero.

We will later see that such transformations form the backbone of algorithms for computing orthogonal symplectic factorizations which motivates us to call matrices of the form

$$E_j(x) := E_j(v,w,\beta,\gamma,\theta) := H_j(v,\beta)G_j(\theta)H_j(w,\gamma) \tag{2.1}$$

elementary (orthogonal symplectic).

Block algorithms for QR factorization, Hessenberg, bidiagonal or tridiagonal reduction implicitly rely on WY representations for products of Householder matrices [7, 10, 16]. Furthermore, the generation of the involved orthogonal transformation matrices can be implemented efficiently by making explicit use of such representations, see for example the LAPACK routine `DORGBR`. Thus, in order to develop block algorithms for orthogonal symplectic factorizations we have to derive a modified WY representation theorem for products of E_j-matrices with E_j as in (2.1). Of course, since G_j and H_j can be written as the product of two Householder matrices, we could apply the standard WY representation to obtain such a representation. However, such an approach would ignore the structures in G_j, H_j and would consequently lead to considerably higher memory and run-time requirements. The following theorem presents a modified representation where these structures are exploited.

THEOREM 2.1. *Let $k \le n$ and $\mathcal{Q} = E_{j_1}(x_1)E_{j_2}(x_2)\dots E_{j_k}(x_k)$, where the matrices $E_{j_i}(x_i)$ are defined as in (2.1) with $j_i \in [1,n]$ and $x_i \in \mathbb{R}^{2n}$. Then there exist matrices $R \in \mathbb{R}^{3k\times k}, S \in \mathbb{R}^{k\times 3k}, T \in \mathbb{R}^{3k\times 3k}$ and $W \in \mathbb{R}^{n\times 3k}$ so that*

$$\mathcal{Q} = \begin{bmatrix} I_n + WTW^T & WRSW^T \\ -WRSW^T & I_n + WTW^T \end{bmatrix}. \tag{2.2}$$

Furthermore, these matrices can be partitioned as

$$R = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix},\ S = \begin{bmatrix} S_1 & S_2 & S_3 \end{bmatrix},\ T = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix},$$

where all matrices $R_i, S_l, T_{il} \in \mathbb{R}^{k\times k}$ are upper triangular, and

$$W = \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix},$$

where $W_1, W_2, W_3 \in \mathbb{R}^{n\times k}$ and W_2 contains in its i-th column e_{j_i}, the j_i-th column of the $n\times n$ identity matrix.

PROOF. The representation (2.2) is proven by induction w.r.t. k. The case $k = 0$ is clear. Let $\mathcal{Q}$ be represented as in (2.2). Consider for $j := j_{k+1}$ the product

$$\tilde{\mathcal{Q}} := \mathcal{Q}E_j(v,w,\beta,\gamma,\theta) = \mathcal{Q}H_j(v,\beta)G_j(\theta)H_j(w,\gamma).$$

We now show how to find a representation for $\tilde{\mathcal{Q}}$. Similar to the construction of the storage-efficient WY representation [16], the first Householder matrix $H_j(v,\beta)$ is incorporated by updating

$$R_1 \leftarrow \begin{bmatrix} R_1 \\ 0 \end{bmatrix},\ S_1 \leftarrow \begin{bmatrix} S_1 & -\beta SW^Tv \end{bmatrix}, \tag{2.3}$$

$$T_{11} \leftarrow \begin{bmatrix} T_{11} & -\beta T_{1,:}W^Tv \\ 0 & -\beta \end{bmatrix},\ T_{i1} \leftarrow \begin{bmatrix} T_{i1} & -\beta T_{i,:}W^Tv \end{bmatrix}, \tag{2.4}$$

$$T_{1l} \leftarrow \begin{bmatrix} T_{1l} \\ 0 \end{bmatrix},\ W_1 \leftarrow \begin{bmatrix} W_1 & v \end{bmatrix}, \tag{2.5}$$

where $i, l \in \{2, 3\}$ and $T_{i,:}$ denotes the i-th block row of T. By straightforward computation it can be shown that the following update yields a representation for $\mathcal{Q}H_j(v, \beta)G_j(\theta)$,

$$R_2 \leftarrow \begin{bmatrix} R_2 & T_{2,:}W^T e_j \\ 0 & 1 \end{bmatrix}, \; R_i \leftarrow \begin{bmatrix} R_i & T_{i,:}W^T e_j \end{bmatrix}, \; S_2 \leftarrow \begin{bmatrix} S_2 & \bar{c}S_2 W^T \\ 0 & -\bar{s} \end{bmatrix},$$

$$S_i \leftarrow \begin{bmatrix} S_i \\ 0 \end{bmatrix}, \; T_{22} \leftarrow \begin{bmatrix} T_{22} & (\bar{s}R_2 S + \bar{c}T_{2,:})W^T e_j \\ 0 & \bar{c} \end{bmatrix},$$

$$T_{i2} \leftarrow \begin{bmatrix} T_{i2} & (\bar{s}R_i S + \bar{c}T_{i,:})W^T e_j \end{bmatrix}, \; T_{2i} \leftarrow \begin{bmatrix} T_{2i} \\ 0 \end{bmatrix}, \; W_2 \leftarrow \begin{bmatrix} W_2 & e_j \end{bmatrix},$$

where $\bar{c} = 1 - \cos\theta$, $\bar{s} = \sin\theta$ and $i, l \in \{1, 3\}$. The second Householder matrix $H_j(w, \gamma)$ is treated similar to (2.3)–(2.5). □

An inspection of the preceding proof reveals that the matrices R_3, S_1, T_{21}, T_{31} and T_{32} are actually strictly upper triangular and the matrix R_2 is unit upper triangular. If $j_i = i$ then the upper $k \times k$ blocks of W_1, W_3 consist of unit lower triangular matrices and W_2 contains the first k columns of the identity matrix. In this case a thorough implementation of the construction given in the proof of Theorem 2.1 requires $(4k-2)kn + \frac{19}{3}k^3 + \frac{1}{2}k^2 + \mathcal{O}(k)$ floating point operations (flops). For the definition of a *flop* see e.g. [12, p. 18]. The application of the WY-like representation (2.2) to a $2n \times q$ matrix requires $(16k(n-k) + 38k - 2)q$ flops using an implementation of the following algorithm which takes care of all the generic structures present in R, S, T and W.

Algorithm 2.2.

Input: *Matrices* $A_1, A_2 \in \mathbb{R}^{n \times q}$*; matrices* $R \in \mathbb{R}^{3k \times k}$*,* $S \in \mathbb{R}^{k \times 3k}$*,* $T \in \mathbb{R}^{3k \times 3k}$*,* $W \in \mathbb{R}^{n \times 3k}$ *representing the orthogonal symplectic matrix* $\mathcal{Q}$ *as described in Theorem 2.1 for* $j_i = i$*.*

Output: *The matrix* $\begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$ *is overwritten with* $\mathcal{Q}^T \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}$*.*

```
$V_1 = A_1^T W, \; V_2 = A_2^T W$
$Y_1 = V_1 T^T + V_2 S^T R^T$
$Y_2 = V_2 T^T + V_1 S^T R^T$
$A_1 \leftarrow A_1 + V_1 Y_1^T, \; A_2 \leftarrow A_2 + V_2 Y_2^T$
```

3 Block symplectic QR factorization

Using the results of the previous section we can now easily derive a block oriented algorithm for computing the symplectic QR factorization. First, let us recall the standard algorithm [8].

Algorithm 3.1.

Input: *A matrix* $\mathcal{A} \in \mathbb{R}^{2m \times n}$ *with* $m \geq n$*.*

Output: *An orthogonal symplectic matrix* $\mathcal{Q} \in \mathbb{R}^{2m \times 2m}$*;* $\mathcal{A}$ *is overwritten with* $\mathcal{R} = \mathcal{Q}^T \mathcal{A}$ *having the form (1.1).*

```
$\mathcal{Q} = I_{2m}$.
FOR $j = 1, \ldots, n$
  Set $x = \mathcal{A}e_j$.
```

```
        Apply Algorithm 2.1 to compute $E_j(x)$.
        Update $\mathcal{A} \leftarrow E_j(x)^T\mathcal{A}$, $\mathcal{Q} \leftarrow \mathcal{Q}E_j(x)$.
    END FOR
```

Let us partition the matrix $\mathcal{A}$ into block columns

$$\mathcal{A} = \left[\begin{array}{cccc} A_1 & A_2 & \dots & A_N \end{array} \right],$$

For convenience only, we will assume that each A_i has n_b columns so that $n = N \cdot n_b$. Our block algorithm for the symplectic QR factorization goes hand in hand with block algorithms for the standard QR factorization [7]. The idea is as follows. At the beginning of step p $(1 \leq p \leq N)$ the matrix $\mathcal{A}$ has been overwritten with

$$\mathcal{Q}_{j-1}\cdots\mathcal{Q}_1\mathcal{A} = \begin{array}{c} \\ {\scriptstyle (p-1)n_b} \\ {\scriptstyle r} \\ {\scriptstyle (p-1)n_b} \\ {\scriptstyle r} \end{array} \begin{array}{c} \begin{array}{ccc} {\scriptstyle (p-1)n_b} & {\scriptstyle n_b} & {\scriptstyle q} \end{array} \\ \left[\begin{array}{ccc} R_{11} & R_{12} & R_{13} \\ 0 & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \\ 0 & R_{42} & R_{43} \end{array} \right] \end{array},$$

where $q = n - pn_b$ and $r = m - (p-1)n_b$. The symplectic QR factorization of $\left[\begin{smallmatrix} R_{22} \\ R_{42} \end{smallmatrix}\right]$ is then computed and the resulting orthogonal symplectic factor applied to $\left[\begin{smallmatrix} R_{23} \\ R_{43} \end{smallmatrix}\right]$. In the following formal description of this procedure the colon notation $A(i_1 : i_2, j_1 : j_2)$ is used to designate the sub-matrix of A defined by rows i_1 through i_2 and columns j_1 through j_2.

Algorithm 3.2.

Input: *A matrix $\mathcal{A} \in \mathbb{R}^{2m\times n}$ with $m \geq n$ and $n = N \cdot n_b$.*

Output: *An orthogonal symplectic matrix $\mathcal{Q} \in \mathbb{R}^{2m\times 2m}$; $\mathcal{A}$ is overwritten with $\mathcal{R} = \mathcal{Q}^T\mathcal{A}$ having the form (1.1). In contrast to Algorithm 3.1 a block oriented method is used.*

```
    $\mathcal{Q} = I_{2m}$.
    FOR $p = 1, \dots, N$
        $s = (p-1)n_b + 1$
        Apply Algorithm 3.1 and the construction given in the proof of
        Theorem 2.1 to compute the $WY$-like representation (2.2) of an
        orthogonal symplectic matrix $\mathcal{Q}_p$ so that

            $\mathcal{Q}_p^T \left[ \begin{array}{c} \mathcal{A}(s:m, s:s+n_b-1) \\ \mathcal{A}(m+s:2m, s-1:s+n_b-1) \end{array} \right]$

        has the form (1.1).
        Update $\left[\begin{smallmatrix} \mathcal{A}(s:m,s+n_b:n) \\ \mathcal{A}(m+s:2m,s+n_b:n) \end{smallmatrix}\right] \leftarrow \mathcal{Q}_p^T \left[\begin{smallmatrix} \mathcal{A}(s:m,s+n_b:n) \\ \mathcal{A}(m+s:2m,s+n_b:n) \end{smallmatrix}\right]$ using Alg. 2.2.
        Update $\left[\begin{smallmatrix} \mathcal{Q}(:,s:m) & \mathcal{Q}(:,m+s:2m) \end{smallmatrix}\right] \leftarrow \left[\begin{smallmatrix} \mathcal{Q}(:,s:m) & \mathcal{Q}(:,m+s:2m) \end{smallmatrix}\right] \mathcal{Q}_p$ using Alg. 2.2.
    END FOR
```

In this algorithm,

$$6(2mn^2 - n^3)/N + 29n^3/(3N^2) + \mathcal{O}(n^2)$$

flops are required to generate the WY-like representations while

$$8(mn^2 - n^3/3) - (8mn^2 - 19n^3)/N - 49n^3/(3N^2) + \mathcal{O}(n^2)$$

flops are necessary to apply them to the matrix $\mathcal{A}$. On the other hand, Algorithm 3.1 requires $8(mn^2 - n^3/3) + \mathcal{O}(n^2)$ flops to compute the factor $\mathcal{R}$. Hence, Algorithm 3.2 is more expensive by roughly a factor of $(1+2.5/N)$, at least when flops are concerned. Basically the same observation holds for the computation of the orthogonal symplectic factor $\mathcal{Q}$. All but needless to remark that in an efficient implementation $\mathcal{Q}$ would be accumulated in reversed order.

4 Block symplectic URV factorization

In the same manner WY-like representations allow us to develop block algorithms for virtually any kind of one-sided orthogonal symplectic factorization. Some new difficulties arise when we consider two-sided factorizations. In this case and in contrast to the symplectic QR factorization it is often impossible to reduce a subset of columns without touching other parts of the matrix. Hence, more effort is necessary to resolve the dependencies between the individual elementary transformations used to construct a two-sided factorization. Let us illuminate this point with the symplectic URV factorization.

ALGORITHM 4.1. *[6]*

Input: *A matrix* $\mathcal{A} \in \mathbb{R}^{2n\times 2n}$.

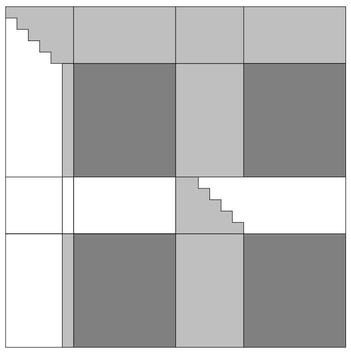
Output: *Orthogonal symplectic matrices* $\mathcal{U}, \mathcal{V} \in \mathbb{R}^{2n\times 2n}$; $\mathcal{A}$ *is overwritten with* $\mathcal{R} = \mathcal{U}^T\mathcal{A}\mathcal{V}$ *having the form (1.2).*

```
$\mathcal{U} = \mathcal{V} = I_{2n}$.
FOR $j = 1, \dots, n$
  Set $x = \mathcal{A}e_j$.
  Apply Algorithm 2.1 to compute $E_j(x)$.
  Update $\tilde{\mathcal{A}} = E_j(x)^T\mathcal{A}$, $\mathcal{U} = \mathcal{U}E_j(x)$.
  IF $j < n$ THEN
    Set $y = \mathcal{A}^T e_{n+j}$.
    Apply Algorithm 2.1 to compute $E_{j+1}(y)$.
    Update $\mathcal{A} \leftarrow \mathcal{A}E_{j+1}(y)$, $\mathcal{V} \leftarrow \mathcal{V}E_{j+1}(y)$.
  END IF
END FOR
```

Let us assume that Algorithm 4.1 is stopped after $k < n$ loops. Denote the so far updated matrix by $\mathcal{A}^{(k)}$ and partition

$$\mathcal{A}^{(k)} = \begin{bmatrix} A^{(k)} & G^{(k)} \\ Q^{(k)} & B^{(k)} \end{bmatrix}, \tag{4.1}$$

where each block is $n \times n$. According to the usual terminology for block algorithms we say that $\mathcal{A}^{(k)}$ is *k-panel reduced*. The matrix $\mathcal{A}^{(k)}$ emerged from $\mathcal{A}^{(0)} = \mathcal{A}$ after k elementary transformations have been applied to both sides of $\mathcal{A}^{(0)}$. Applying Theorem 2.2 to these transformations and multiplying $\mathcal{A}^{(0)}$



□ $\tilde{\mathcal{A}}^{(k)}(i,j) = 0$

▨ $\tilde{\mathcal{A}}^{(k)}(i,j) = \mathcal{A}^{(k)}(i,j)$

■ $\tilde{\mathcal{A}}^{(k)}(i,j) = \mathcal{A}^{(0)}(i,j)$

Figure 4.1: Structure of $\tilde{\mathcal{A}}^{(k)}$ for $k = 5, n = 15$. White and pale-gray parts contain the reduced k-panel, these are elements of the matrix $\mathcal{A}^{(k)}$. Dark-gray parts contain elements of the original matrix $\mathcal{A}^{(0)}$.

from both sides by the corresponding WY-like representations show that there exist $n \times 3k$ matrices $\tilde{U}, \tilde{V}, \tilde{X}_{\{A,B,G,Q\}}, \tilde{Y}_{\{A,B,G,Q\}}$ so that

$$\mathcal{A}^{(k)} = \left[\begin{array}{cc} A^{(0)} + \tilde{U}\tilde{X}_A^T + \tilde{Y}_A\tilde{V}^T & G^{(0)} + \tilde{U}\tilde{X}_G^T + \tilde{Y}_G\tilde{V}^T \\ Q^{(0)} + \tilde{U}\tilde{X}_Q^T + \tilde{Y}_Q\tilde{V}^T & B^{(0)} + \tilde{U}\tilde{X}_B^T + \tilde{Y}_B\tilde{V}^T \end{array} \right]. \tag{4.2}$$

Clearly, the above representation of $\mathcal{A}^{(k)}$ would directly lead to a block version of Algorithm 4.1. Unfortunately, things are not that simple because the definition of the elementary transformations used in Algorithm 4.1 requires that columns $1 : k$ and rows $n{+}1 : n{+}k$ are updated. Also, the first instruction in loop $k{+}1$ would require that the $(k+1)$-th column of $\mathcal{A}^{(k)}$ is known at this time. We therefore remove the parts from $\tilde{U}, \tilde{V}, \tilde{X}_{\{A,B,G,Q\}}$ and $\tilde{Y}_{\{A,B,G,Q\}}$ that correspond to these portions of the matrix $\mathcal{A}^{(k)}$. In turn, the matrices $A^{(0)}, B^{(0)}, G^{(0)}, Q^{(0)}$ in (4.2) must be altered to compensate these removals. Let $\tilde{\mathcal{A}}^{(k)}$ be equal to $\mathcal{A}^{(0)}$ with columns $1 : k+1$, $n+1 : n+k+1$ and rows $1 : k$, $n+1 : n+k$ superseded by the corresponding entries of $\mathcal{A}^{(k)}$ as illustrated in Figure 4.1. Furthermore, $\tilde{\mathcal{A}}^{(k)}$ is partitioned into blocks $\tilde{A}^{(k)}, \tilde{B}^{(k)}, \tilde{G}^{(k)}$ and $\tilde{Q}^{(k)}$ similarly to (4.1).

Altogether, we consider the modified representation

$$\mathcal{A}^{(k)} = \left[\begin{array}{cc} \tilde{A}^{(k)} + UX_A^T + Y_AV^T & \tilde{G}^{(k)} + UX_G^T + Y_GV^T \\ \tilde{Q}^{(k)} + UX_Q^T + Y_QV^T & \tilde{B}^{(k)} + UX_B^T + Y_BV^T \end{array} \right], \tag{4.3}$$

where $U, V, X_{\{A,B,G,Q\}}, Y_{\{A,B,G,Q\}}$ have been reduced to $n \times 2k$ matrices.

We now show how to pass from (4.3) to an analogous representation for $\mathcal{A}^{(k+1)}$. In the following algorithm the symbol '$\star$' denotes a placeholder which may take any value in the set $\{A, B, G, Q\}$.

ALGORITHM 4.2.

Input: *A k-panel reduced matrix $\mathcal{A}^{(k)} \in \mathbb{R}^{2n \times 2n}$ represented as in (4.3).*

Output: *A representation of the form (4.3) for the $(k+1)$-panel reduced matrix $\mathcal{A}^{(k+1)}$.*

```
    % Incorporate transformations from the left.
    Apply Algorithm 2.1 to compute $E_{k+1}(\tilde{\mathcal{A}}^{(k)}e_{k+1}) = E_{k+1}(v, w, \beta, \gamma, \theta)$ and
    update the $(k+1)$-th column of $\tilde{\mathcal{A}}^{(k)}$.
    FOR EACH $\star \in \{A, B, G, Q\}$ DO
      $X_\star \leftarrow [X_\star, -\beta((\tilde{\star}^{(k)})^T v + X_\star U^T v + V Y_\star^T v)]$, $U \leftarrow [U, v]$
      $\tilde{\star}^{(k)}(k+1,:) \leftarrow \tilde{\star}^{(k)}(k+1,:) + X_\star(k+1,:)U^T + V(k+1,:)Y_\star^T$
      $X_\star(k+1,:) = 0$, $V(k+1,:) = 0$
    END FOR
    $\tilde{\mathcal{A}}^{(k)} \leftarrow G_{k+1}(\theta)\tilde{\mathcal{A}}^{(k)}$
    FOR EACH $\star \in \{A, B, G, Q\}$ DO
      $w(k+1) = 0$, $x_\star = -\gamma((\tilde{\star}^{(k)})^T w + X_\star U^T w + V Y_\star^T w)$
      $X_\star \leftarrow [X_\star, x_\star]$, $U \leftarrow [U, w]$
      $\tilde{\star}^{(k)}(k+1,:) \leftarrow \tilde{\star}^{(k)}(k+1,:) + x_\star^T$
    END FOR
    % Incorporate transformations from the right.
    Apply Algorithm 2.1 to compute $E_{k+2}((\tilde{\mathcal{A}}^{(k)})^T e_{n+k+1}) = E_{k+2}(v, w, \beta, \gamma, \theta)$
    and update the $(n+k+1)$-th row of $\tilde{\mathcal{A}}^{(k)}$.
    FOR EACH $\star \in \{A, B, G, Q\}$ DO
      $Y_\star \leftarrow [Y_\star, -\beta(\tilde{\star}^{(k)} v + U X_\star^T v + Y_\star V^T v)]$, $V \leftarrow [V, v]$
      $\tilde{\star}^{(k)}(:,k+2) \leftarrow \tilde{\star}^{(k)}(:,k+2) + X_\star U(k+2,:)^T + V Y_\star(k+2,:)^T$
      $U(k+2,:) = 0$, $Y_\star(k+2,:) = 0$
    END FOR
    $\tilde{\mathcal{A}}^{(k)} \leftarrow \tilde{\mathcal{A}}^{(k)} G_{k+2}(\theta)$
    FOR EACH $\star \in \{A, B, G, Q\}$ DO
      $w(k+2) = 0$, $y_\star = -\gamma(\tilde{\star}^{(k)} w + U X_\star^T w + Y_\star V^T w)$
      $Y_\star \leftarrow [Y_\star, y_\star]$, $V \leftarrow [V, w]$
      $\tilde{\star}^{(k)}(:,k+2) \leftarrow \tilde{\star}^{(k)}(:,k+2) + y_\star$
    END FOR
    $\tilde{\mathcal{A}}^{(k+1)} = \tilde{\mathcal{A}}^{(k)}$
```

Subsequent application of Algorithm 4.2 yields representation (4.3) requiring

$$16 \cdot (2kn^2 + 7k^2n - 13k^3/3) + 42kn + \mathcal{O}(k^2)$$

flops.

The rest of the story is easily told. Using (4.3) the matrix $\mathcal{A}^{(k)}$ is computed via eight rank-$2k$ updates of order $n-k$. The next panel to be reduced resides in rows and columns $k+1 : 2k$, $n+k+1 : n+2k$ of $\mathcal{A}^{(k)}$ as illustrated in Figure 4.2. Algorithm 4.2 is repeatedly applied to the matrix

$$\left[\begin{array}{cc} A^{(k)}(k+1:n, k+1:n) & G^{(k)}(k+1:n, k+1:n) \\ Q^{(k)}(k+1:n, k+1:n) & B^{(k)}(k+1:n, k+1:n) \end{array}\right]$$

to reduce its leading k-panel. Again, eight rank-$2k$ updates, now of order $n-2k$, yield $\star^{(2k)}(k+1:n, k+1:n)$ for $\star \in \{A, B, G, Q\}$. It remains to update rows $1:k$ and columns $n+1:n+k+1$ of $\mathcal{A}^{(k)}$. This could be achieved by applying WY-like representations of the orthogonal symplectic transformations involved in the reduction of the second panel. In our implementation, however,

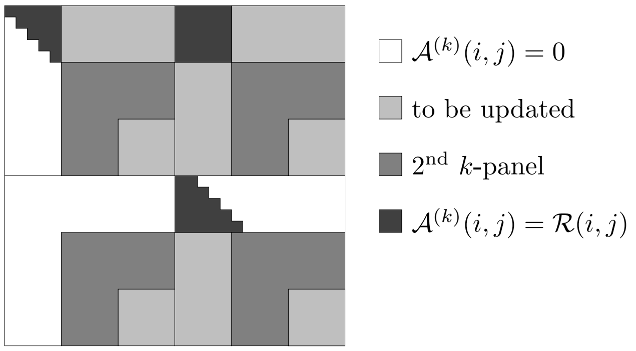

$\mathcal{A}^{(k)}(i,j) = 0$
to be updated
2nd k-panel
$\mathcal{A}^{(k)}(i,j) = \mathcal{R}(i,j)$

Figure 4.2: Reduction of the second k-panel for $k = 5, n = 15$. White and black parts partly contain the first k-panel and are not affected by subsequent panel reductions. Dark-gray parts contain the second k-panel and pale-gray parts must be updated after the second panel has been reduced.

we include these parts in Algorithm 4.2 so that the subsequent rank-$2k$ updates readily yield the matrix $\mathcal{A}^{(2k)}$. For more details the reader is referred to the Fortran implementation of this algorithm [18].

Assume that $n = N \cdot k$, then the procedure described above requires

$$80n^3/3 + 64n^3/N - 16n^3/N^2 + \mathcal{O}(n^2)$$

flops to compute the R-factor in the symplectic URV-factorization of a $2n \times 2n$ matrix. Since the unblocked version, Algorithm 4.1, requires $80n^3/3 + \mathcal{O}(n^2)$ flops for the same task, we see that blocking is more expensive by a factor of $(1 + 2.4/N)$.

5 Numerical Stability

The derived block algorithms would be unedifying if they flawed the favorable error analysis of orthogonal factorizations. To show backward stability for the construction and the application of the WY-like representation in Section 2 we use techniques described in the book by Higham [13]. First, let us establish the following inequalities.

LEMMA 5.1. *Let $\hat{R}, \hat{S}, \hat{T}$ and $\hat{W}$ be the computed factors of the block representation constructed as in the proof of Theorem 2.1 and set*

$$\hat{\mathcal{Q}} = \begin{bmatrix} I + \hat{W}\hat{T}\hat{W}^T & \hat{W}\hat{R}\hat{S}\hat{W}^T \\ -\hat{W}\hat{R}\hat{S}\hat{W}^T & I + \hat{W}\hat{T}\hat{W}^T \end{bmatrix}.$$

Let $\mathbf{u}$ denote the unit roundoff, then

$$\|\hat{\mathcal{Q}}^T\hat{\mathcal{Q}} - I\|_2 \le \mathbf{u}d_{\mathcal{Q}}, \tag{5.1}$$

$$\|\hat{R}\|_2 \le d_R, \quad \|\hat{S}\|_2 \le d_S, \quad \|\hat{T}\|_2 \le d_T, \quad \|\hat{W}\|_2 \le d_W, \tag{5.2}$$

for modest constants $d_{\mathcal{Q}}, d_R, d_S, d_T$ and d_W.

Proof. Inequality (5.1) is shown by induction. For the evaluation of $\hat{\mathcal{Q}}_1$ with $\mathcal{Q}_1 := \mathcal{Q}H_j(v,\beta)$ we may assume that $|\hat{v} - v| \le \gamma_{cn}|v|$, $\|v\|_2 = \sqrt{2}$, and $\beta = 1$, where the quantity γ_{cn} denotes $\mathbf{u}cn/(1-\mathbf{u}cn)$ with a small constant $c > 1$. Then, using formulas (2.3)–(2.4) and Lemma 18.3 [13] on the backward error in products of Householder matrices, it follows that

$$\|\hat{\mathcal{Q}}_1^T\hat{\mathcal{Q}}_1 - I\|_2 \le \mathbf{u}d_{\mathcal{Q}} + \sqrt{n}\gamma_{cn} =: \mathbf{u}d'_{\mathcal{Q}}. \tag{5.3}$$

Similarly, $|\cos\hat{\theta} - \cos\theta| + |\sin\hat{\theta} - \sin\theta| \le \gamma_{c'}$ with a small constant $c' > 0$. The factored matrix $\hat{\mathcal{Q}}_2$ with $\mathcal{Q}_2 := \mathcal{Q}_1 G_j(\theta)$ satisfies

$$\|\hat{\mathcal{Q}}_2^T\hat{\mathcal{Q}}_2 - I\|_2 \le \mathbf{u}d'_{\mathcal{Q}} + \sqrt{n}\gamma_c.$$

Repeated application of (5.3) to $\mathcal{Q}_3 := \mathcal{Q}_2 H_j(w,\gamma)$ proves (5.1). The inequalities in (5.2) readily follow from the construction of R, S, T and W. ◻

Inequality (5.1) implies that the matrix $\hat{\mathcal{Q}}$ is close to orthogonality. Closeness to symplecticity is shown by the following lemma.

Lemma 5.2. *Let $\hat{\mathcal{Q}} = \begin{bmatrix} C & S \\ -S & C \end{bmatrix}$ be invertible with $C, S \in \mathbb{R}^{n\times n}$, then there exist an orthogonal symplectic matrix $\mathcal{U}$ and a symmetric matrix $\mathcal{H}$ such that $\mathcal{Q} = \mathcal{U}\mathcal{H}$ and*

$$\frac{\|\hat{\mathcal{Q}}^T\hat{\mathcal{Q}} - I\|_2}{\|\hat{\mathcal{Q}}\|_2 + 1} \le \|\hat{\mathcal{Q}} - \mathcal{U}\|_2 \le \|\hat{\mathcal{Q}}^T\hat{\mathcal{Q}} - I\|_2. \tag{5.4}$$

Proof. The matrix $\hat{\mathcal{Q}}^T\hat{\mathcal{Q}}$ is symmetric and skew-Hamiltonian, i.e. $\hat{\mathcal{Q}}^T\hat{\mathcal{Q}}J = -J^T\hat{\mathcal{Q}}^T\hat{\mathcal{Q}}$. Hence, there exists an orthogonal symplectic matrix $\mathcal{U}_1$ such that $\mathcal{D} = \mathcal{U}_1^T\hat{\mathcal{Q}}^T\hat{\mathcal{Q}}\mathcal{U}_1$ is diagonal [15]. Similarly, there exists an orthogonal symplectic matrix $\mathcal{U}_2$ with $\mathcal{D} = \mathcal{U}_2^T\hat{\mathcal{Q}}\hat{\mathcal{Q}}^T\mathcal{U}_2$. The invertibility of $\mathcal{D}$ implies that $\hat{\mathcal{D}} = \mathcal{U}_2^T\hat{\mathcal{Q}}\mathcal{U}_1$ is diagonal. The first part of the lemma now follows by setting $\mathcal{U} = \mathcal{U}_2\mathcal{U}_1^T$ and $\mathcal{H} = \mathcal{U}_1\hat{\mathcal{D}}\mathcal{U}_1^T$. Inequality (5.4) is a well-known result, see e.g. [13, p.389]. ◻

Lemma 5.1 shows that under the usual assumptions on matrix multiplication, the forward errors of the computed matrices $\hat{B}_1$ and $\hat{B}_2$ in Algorithm 2.2 satisfy

$$\begin{aligned}
\left\|\hat{B}_1 - B_1\right\|_2 &\le \mathbf{u}\|A_1\|_2 + \mathbf{u}d_W^2[c_T(k,n)\|A_1\|_2 + c_R(k,n)\|A_2\|_2], \\
\left\|\hat{B}_2 - B_2\right\|_2 &\le \mathbf{u}\|A_2\|_2 + \mathbf{u}d_W^2[c_T(k,n)\|A_2\|_2 + c_R(k,n)\|A_1\|_2],
\end{aligned}$$

where

$$c_T(k,n) := d_T(18k^2 + n^2 + 2), \quad c_R(k,n) := d_R d_S(19k^2 + n^2 + 2).$$

Lemma 5.2 together with inequality (5.1) imply the existence of an orthogonal symplectic matrix $\mathcal{U}$ so that $\hat{\mathcal{Q}} = \mathcal{U} + \triangle\mathcal{U}$ with $\|\triangle\mathcal{U}\|_2 \le \mathbf{u}d_{\mathcal{Q}}$. This enables us to bound the backward error of Algorithm 2.2,

$$\begin{bmatrix} \hat{B}_1 \\ \hat{B}_2 \end{bmatrix} = \mathcal{U}\begin{bmatrix} \hat{A}_1 + \triangle A_1 \\ \hat{A}_2 + \triangle A_2 \end{bmatrix},$$

where

$$\left\| \begin{bmatrix} \triangle A_1 \\ \triangle A_2 \end{bmatrix} \right\|_2 \leq \sqrt{2}\mathbf{u} \left[1 + d_Q + d_W^2 (c_T(k,n) + c_R(k,n)) \right] \left\| \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \right\|_2 .$$

This immediately verifies numerical backward stability for symplectic QR factorizations constructed as described in Algorithm 3.2. The analysis of the block algorithm for symplectic URV factorizations presented in Section 4 is complicated by the fact that parts from the WY-like representations for the left and right elementary transformations are removed to keep the k-panel updated. However, it can easily be shown that these removals do not introduce numerical instabilities in the symplectic URV factorization.

6 Numerical Results

The described block algorithms are implemented in Fortran 77 in accordance to the SLICOT implementation and documentation standards [17]. They form an integral part of a prospective software library for computing eigenvalue problems with Hamiltonian, symplectic or block cyclic structures [18].

To demonstrate the efficiency of these implementations we present numerical examples run on an Origin2000 computer equipped with 400MHz IP27 R12000 processors and sixteen gigabytes of memory. The implementations were compiled with version 7.30 of the MIPSpro Fortran 77 compiler with options `-64 TARG:platform=ip27 -Ofa st=ip27 -LNO`. The programs called optimized BLAS and LAPACK subroutines from the SGI/Cray Scientific Library version 1.2.0.0. Timings were carried out using matrices with pseudorandom entries uniformly distributed in the interval $[-1, 1]$. Unblocked code was used for all subproblems with column or row dimension smaller than 65 (`NX = 64`). Each matrix was stored in an array with leading dimension slightly larger than the number of rows to avoid unnecessary cache conflicts.

Table 6.1 shows the result for `DGESQB`, an implementation of Algorithm 3.1. Rows with block size $n_b = 1$ correspond to the unblocked variant of this algorithm. Columns with heading $\mathcal{R}$ show timings when only the reduced matrix $\mathcal{R}$ was computed. Additional times necessary to generate the orthogonal symplectic factor $\mathcal{Q}$ are displayed in columns with heading $\mathcal{Q}$. The results show that the block algorithm outperforms the unblocked one for all chosen matrix dimensions under the assumption that a suitable value for the block size n_b has been used. The best improvement has been obtained for $m = 1024, n = 1024$ where the block algorithm saved 74.8% of the execution time for the computation of both factors.

The results for `DGESUB`, an implementation of the block algorithm described in Section 4, are displayed in Table 6.2. Again, the column with heading $\mathcal{R}$ refers to timings when only the reduced matrix $\mathcal{R}$ was computed. The additional times for generating the factors $\mathcal{U}$ and $\mathcal{V}$ are displayed in columns four and five, respectively. Albeit not so dramatic as for the symplectic QR factorization the results show considerable improvements when the matrix order is sufficiently large. At its best, the block algorithm saved 47.6% of the execution time when

`DGESQB`		$n = 128$		$n = 256$		$n = 512$		$n = 1024$	
m	n_b	$\mathcal{R}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{Q}$	$\mathcal{R}$	$\mathcal{Q}$
128	1	0.10	0.10	0.26	0.10	0.75	0.10	2.71	0.10
128	8	*0.08*	*0.08*	*0.21*	*0.08*	*0.50*	*0.08*	1.54	*0.08*
128	16	0.09	*0.08*	0.22	*0.08*	0.51	*0.08*	1.52	*0.08*
128	24	0.10	0.09	0.23	0.09	0.52	0.09	*1.50*	0.09
128	32	0.10	0.10	0.25	0.10	0.56	0.10	1.65	0.10
256	1	0.24	0.24	0.88	0.88	3.32	0.88	9.71	0.90
256	8	*0.18*	*0.18*	*0.54*	0.54	1.46	0.54	3.87	0.54
256	16	*0.18*	*0.18*	*0.54*	*0.50*	*1.42*	*0.50*	*3.63*	*0.51*
256	24	0.20	0.19	0.57	0.53	1.46	0.53	3.72	0.53
256	32	0.20	0.20	0.61	0.56	1.52	0.56	3.93	0.56
512	1	0.59	0.60	3.25	3.27	13.10	13.16	35.16	13.16
512	16	0.40	*0.39*	*1.31*	1.28	4.33	4.20	11.34	4.15
512	24	*0.39*	*0.39*	*1.31*	*1.27*	4.26	4.11	11.04	4.07
512	32	0.42	0.41	*1.31*	*1.27*	*4.17*	*3.96*	*10.78*	*3.94*
512	48	0.46	0.45	1.40	1.35	4.26	4.06	11.05	4.04
1024	1	1.86	1.89	9.15	9.20	34.99	35.11	113.29	113.58
1024	16	0.87	0.88	2.94	2.92	10.43	10.21	33.85	33.17
1024	24	*0.85*	*0.84*	2.89	2.85	9.84	9.71	31.90	31.06
1024	32	0.89	0.89	*2.81*	*2.77*	*9.29*	*9.13*	29.68	28.65
1024	48	0.97	0.96	2.95	2.92	9.34	9.15	*29.17*	*27.93*

Table 6.1: Performance results in seconds for the symplectic QR factorization of an $m \times n$ matrix.

the complete symplectic URV factorization of a 2048×2048 ($n = 1024$) matrix was computed.

The accuracy of the block algorithms has been tested for various random matrices as well as Hamiltonian matrices obtained from the Riccati benchmark collection [4]. We measured orthogonality of the factors $\mathcal{Q}, \mathcal{U}$, $\mathcal{V}$ and the relative residuals $\|\mathcal{QR} - \mathcal{A}\|_1 / \|\mathcal{A}\|_1$, $\|\mathcal{URV} - \mathcal{A}\|_1 / \|\mathcal{A}\|_1$. The results for the standard algorithms and the new block algorithms are qualitatively the same.

7 Final Remark

While the symplectic QR factorization basically covers the range of practically important one-sided orthogonal symplectic factorizations an important class of two-sided factorizations, so called PVL reductions [15], is not considered in this work. The PVL reduction of a Hamiltonian matrix is extensively used in the OSMARE algorithm [1]. However, with increasing matrix dimensions, OSMARE

`DGESUB`				
n	n_b	$\mathcal{R}$	$\mathcal{U}$	$\mathcal{V}$
128	1	0.53	0.11	0.11
128	8	*0.48*	*0.08*	*0.09*
128	16	0.52	*0.08*	*0.09*
256	1	6.91	0.87	0.89
256	8	*4.74*	*0.50*	*0.52*
256	16	5.12	*0.50*	0.53
256	32	5.82	0.55	0.58
512	1	66.79	13.04	12.90
512	8	42.17	4.82	5.15
512	16	*42.05*	4.07	4.34
512	32	44.02	*3.88*	*4.11*
1024	1	563.16	113.40	114.02
1024	16	377.55	32.52	33.42
1024	32	*318.84*	*28.13*	*29.29*
1024	64	350.11	28.98	30.32

Table 6.2: Performance results in seconds for the symplectic URV factorization of an $n \times n$ matrix.

heavily suffers from forward instabilities in the QR algorithm. The PVL reduction of a skew-Hamiltonian matrix constitutes a preprocessing step for eigenvalue computations [15]. Developing an efficient block algorithm for this reduction would require to have an efficient BLAS for skew-symmetric block updates of the form $C \leftarrow C + AB^T - BA^T$ handy. Unfortunately, such a subroutine is not yet defined in the BLAS standard.

8 Acknowledgments

The author sincerely thanks Peter Benner, Erik Elmroth and Christian Mehl for various useful improvements and corrections to earlier versions of this paper. Special thanks go to his (former) supervisors Ralph Byers and Volker Mehrmann.

REFERENCES

1. G. Ammar, P. Benner, and V. Mehrmann. A multishift algorithm for the numerical solution of algebraic Riccati equations. *Electron. Trans. Numer. Anal.*, 1(Sept.):33–48 (electronic only), 1993.
2. 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. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

3. P. Benner, R. Byers, V. Mehrmann, and H. Xu. Numerical computation of deflating subspaces of skew-Hamiltonian/Hamiltonian pencils. *SIAM J. Matrix Anal. Appl.*, 24(1):165–190, 2002.
4. P. Benner, A.J. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case. Technical Report SPC 95_22, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.
5. P. Benner, V. Mehrmann, and H. Xu. A new method for computing the stable invariant subspace of a real Hamiltonian matrix. *J. Comput. Appl. Math.*, 86:17–43, 1997.
6. P. Benner, V. Mehrmann, and H. Xu. A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils. *Numer. Math.*, 78(3):329–358, 1998.
7. C. Bischof and C. Van Loan. The *WY* representation for products of Householder matrices. *SIAM J. Sci. Statist. Comput.*, 8(1):S2–S13, 1987. Parallel processing for scientific computing (Norfolk, Va., 1985).
8. A. Bunse-Gerstner. Matrix factorizations for symplectic *QR*-like methods. *Linear Algebra Appl.*, 83:49–77, 1986.
9. J. J. Dongarra, D. C. Sorensen, and S. J. Hammarling. Block reduction of matrices to condensed forms for eigenvalue computations. *J. Comput. Appl. Math.*, 27(1-2):215–227, 1989. Reprinted in *Parallel algorithms for numerical linear algebra*, 215–227, North-Holland, Amsterdam, 1990.
10. E. Elmroth and F. Gustavson. Applying recursion to serial and parallel *QR* factorization leads to better performance. *IBM J. Research & Development*, 44(4):605–624, 2000.
11. G. Freiling, V. Mehrmann, and H. Xu. Existence, uniqueness, and parametrization of Lagrangian invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 23(4):1045–1069, 2002.
12. G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, third edition, 1996.
13. N. J. Higham. *Accuracy and stability of numerical algorithms.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1996.
14. B. J. Leimkuhler and E. S. Van Vleck. Orthosymplectic integration of linear Hamiltonian systems. *Numer. Math.*, 77(2):269–282, 1997.
15. C. Paige and C. Van Loan. A Schur decomposition for Hamiltonian matrices. *Linear Algebra Appl.*, 41:11–32, 1981.
16. R. Schreiber and C. Van Loan. A storage-efficient *WY* representation for products of Householder transformations. *SIAM J. Sci. Statist. Comput.*, 10(1):53–57, 1989.

17. The Working Group on Software: WGS. *SLICOT Implementation and Documentation Standards 2.1*, 1996. Available from `http://www.win.tue.nl/wgs/reports.html` as WGS-report 96-1.

18. See `http://www.math.tu-berlin.de/~kressner/syperham/`.