Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices I: The Square-Reduced Method

Peter Benner *
Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik
Universität Bremen
28334 Bremen (FRG)
E-mail: benner@math.uni-bremen.de

Ralph Byers †
Department of Mathematics
University of Kansas
405 Snow Hall
Lawrence, KS 66045 (USA)
E-mail: byers@math.ukans.edu

Eric Barth ‡
Department of Mathematics
Kalamazoo College
1200 Academy Street
Kalamazoo, MI 49006 (USA)
E-mail: barth@kzoo.edu

Abstract

This paper describes LAPACK-based Fortran 77 subroutines for the reduction of a Hamiltonian matrix to square-reduced form and the approximation of all its eigenvalues using the implicit version of Van Loan's method. The transformation of the Hamiltonian matrix to a square-reduced form transforms a Hamiltonian eigenvalue problem of order $2n$ to a Hessenberg eigenvalue problem of order n. The eigenvalues of the Hamiltonian matrix are the square roots of those of the Hessenberg matrix. Symplectic scaling and norm scaling are provided, which, in some cases, improve the accuracy of the computed eigenvalues. We demonstrate the performance of the subroutines for several examples and show how they can be used to solve some control-theoretic problems.

Key words: eigenvalues, (square-reduced) Hamiltonian matrix, skew-Hamiltonian matrix, algebraic Riccati equation

AMS(MOS) subject classifications: 65F15, 93B40

---

*Some of this work was complete while this author was with the *Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz (FRG)*. Support was received from *Deutsche Forschungsgemeinschaft*, research grant *Me 790/7-1 Singuläre Steuerungsprobleme*.

†This author was partially supported by National Science Foundation awards CCR-9404425, DMS-9205538, DMS-9628626, CCR-9732671, and by the NSF EPSCoR/K*STAR program through the Center for Advanced Scientific Computing.

‡This author's part of the work was complete while he was with the University of Kansas. Partial support was received from the Kansas Institute for Theoretical and Computational Science.

1 Introduction

This paper describes LAPACK-based Fortran 77 subroutines for the reduction of a Hamiltonian matrix to square-reduced form and the approximation of all its eigenvalues using the implicit version of Van Loan's method [39]. Many applications in linear quadratic and $\mathcal{H}_\infty$–control theory require the computation of the eigenvalues of Hamiltonian matrices

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \tag{1}$$

where $A, G, Q \in \mathbb{R}^{n\times n}$ and G, Q are symmetric. These include the computation of the $\mathcal{H}_\infty$–norm of transfer matrices, see, e.g., [20, 9, 8, 10, 38] and the references given therein, and the problem of calculating the stability radius of a matrix [16, 37]. Hamiltonian matrices are also closely related to continuous-time algebraic Riccati equations (CARE) of the form

$$0 = Q + A^T X + XA - XGX \tag{2}$$

with A, G, Q as in (1) and $X \in \mathbb{R}^{n\times n}$ is a symmetric solution matrix. Often, a *stabilizing* solution X of (2) is required in the sense that all the eigenvalues of $A - GX$ are in the open left half plane. If such a solution exists, these eigenvalues are exactly the *stable* eigenvalues (i.e., those with negative real part) of the Hamiltonian matrix H in (1). If the Hamiltonian matrix H has no eigenvalue with zero real part, then there exists an n-dimensional H-invariant subspace corresponding to the n stable eigenvalues of H which is called the *stable invariant subspace* of H. If this subspace is spanned by the columns of $[\,V^T,\ W^T\,]^T$, $V, W \in \mathbb{R}^{n\times n}$, and V is invertible, then a stabilizing solution X of (2) is given by $X = -WV^{-1}$. For a detailed discussion of the relations of Hamiltonian matrices and continuous-time algebraic Riccati equations as well as properties of solutions of the CARE (2) we refer to [25].

Knowledge of approximations to the eigenvalues of the corresponding Hamiltonian matrix is crucial for some numerical methods for the CARE (2), e.g., for the multishift QR–like methods proposed in [2, 3, 34] and the algorithm presented in [32]. As suggested in [35], they can significantly improve the convergence of the SR algorithm [12] when employed as shifts.

Van Loan's method uses the *square-reduced form* of a Hamiltonian matrix which will be introduced in Section 2. Square-reduced Hamiltonian matrices are used by themselves in the CARE solution methods given in [42, 43].

A *Hamiltonian* matrix is defined by the property $HJ = (HJ)^T$ where

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{3}$$

It is easy to see that $(HJ) = (HJ)^T$ if and only if H is in the form of (1). The eigenvalues of a Hamiltonian matrix can be computed using the QR algorithm for general matrices [23, §7.5]. The Hamiltonian structure of (1), however, can be exploited to reduce memory requirements and arithmetic operations and/or to improve accuracy. Symplectic similarity transformations preserve Hamiltonian structure. A matrix $S \in \mathbb{R}^{2n\times 2n}$ is *symplectic* if $S^T JS = J$ with J as in (3). To preserve numerical stability, it is desirable that similarity transformations be orthogonal as well as symplectic. So far, no algorithm has been found that computes the eigenvalues of a general Hamiltonian matrix by using only orthogonal symplectic similarity transformations. For the special case that $\operatorname{rank} G = 1$ or $\operatorname{rank} Q = 1$, a Hamiltonian QR algorithm was presented in [15]. A structure preserving QR-like algorithm using SR decompositions has been presented in [12, 13, 35]. However, this algorithm suffers from numerical instabilities because it uses symplectic similarity transformations that can not be guaranteed to be bounded in norm. Recently, an orthogonal symplectic method for computing the eigenvalues of a Hamiltonian matrix has been proposed [7]. This method is also based on the square-reduced form of a Hamiltonian matrix but uses non-similarity transformations and does not compute this form explicitly. The method is numerically backward stable at the price of a higher computational cost and higher workspace requirements compared to Van Loan's method. The implementation of this method will be described in [5].

The obstacle to the general use of the Hamiltonian QR algorithm as presented in [15] is the difficulty in reducing a general Hamiltonian matrix to a Hessenberg–like matrix with Hamiltonian structure which is then preserved during the QR iteration. A Hamiltonian matrix H is said to be in *Hamiltonian Hessenberg form* if

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} = \begin{bmatrix} \diagdown & \square \\ * & \diagdown \end{bmatrix},$$

i.e., if A is upper Hessenberg, G is symmetric, and Q is a diagonal matrix with a nonzero entry only in its last diagonal element. In [15] it is shown that this form can be preserved under a Hamiltonian QR iteration. But in [3], Ammar and Mehrmann prove that the first column of an orthogonal matrix that reduces a general Hamiltonian matrix to a Hamiltonian Hessenberg matrix must satisfy a homogeneous system of n quadratic equations. This fact casts doubt on the existence of a mathematically finite procedure for reducing a Hamiltonian matrix to Hamiltonian Hessenberg form.

The method proposed by Van Loan [39] uses the properties of the square of a Hamiltonian matrix. A squared Hamiltonian matrix can be reduced to a 2×2 block triangular matrix with Hessenberg blocks on the diagonal. It is possible to do the reduction by structure-preserving orthogonal, symplectic similarity transformations. So, the eigenvalues of the squared Hamiltonian matrix are computed by a strongly backward stable method.[1] As the structure is fully exploited, only about three quarters of the work space and one third of the computational cost of the general nonsymmetric QR algorithm is required. Unfortunately, by taking the square roots of the eigenvalues of the squared Hamiltonian matrix a loss of accuracy of $O(\sqrt{\varepsilon})$ is possible for tiny eigenvalues (if any). (Here, ε denotes the machine precision.) In [39], it is shown that the eigenvalues computed by this method are exact eigenvalues of a Hamiltonian matrix $H + E$ where $\|E\| \leq O(\sqrt{\varepsilon})\|H\|$.

We use the following notation. Let $A \in \mathbb{R}^{n\times n}$. By $\sigma(A)$ we denote the set of eigenvalues or spectrum of a matrix A. The spectral norm of a matrix is

$$\|A\|_2 = \sqrt{\max\{|\lambda| \;:\; \lambda \in \sigma(A^T A)\}}$$

and the *Frobenius* norm of A is

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{n} a_{ij}^2}.$$

[1] A numerical algorithm is called *(numerically) backward stable* if the computed solution is the exact solution for slightly perturbed initial data. It is said to be *strongly backward stable* if it is backward stable and the perturbations have the same structure as the initial data [11].

The notation $\mathbb{C}^-$, $\imath\mathbb{R}$, and $\mathbb{C}^+$, respectively, where $\imath = \sqrt{-1}$, corresponds to the partitioning of the complex plane into the open left half plane, the imaginary axis, and the open right half plane, respectively. The identity matrix of order n is denoted by I_n and the kth unit vector is denoted e_k. We use the colon notation to specify submatrices. (This notation is also used in [23].) For example, $A_{k:\ell,p:q}$ denotes the submatrix of A defined by its entries a_{ij}, $k \le i \le \ell$, $p \le j \le q$.

In the next section we review Van Loan's square-reduced method. In Section 3 we introduce some scaling strategies which can in some cases improve the accuracy of the computed eigenvalues and describe the details of the implementation. Numerical examples demonstrating the performance of the method are presented in Section 4. In Section 5 we also demonstrate how to use our subroutines for solving two problems from control theory: the distance-to-instability problem and computing the $\mathcal{H}_\infty$–norm of a transfer matrix. Conclusions are drawn in Section 6.

2 Van Loan's Square-Reduced Method

2.1 Algebraic Properties of Hamiltonian and Symplectic Matrices

First we review some well-known results about Hamiltonian and symplectic matrices which form the basis for Van Loan's method. These results can be found in [39], so we omit the proofs.

The spectrum of a Hamiltonian matrix has the following property.

Proposition 1 *Let $\lambda \in \sigma(H)$ where $H \in \mathbb{R}^{2n\times 2n}$ is a Hamiltonian matrix. If $\lambda \in \mathbb{R}$ or $\lambda \in \imath\mathbb{R}$, then $-\lambda \in \sigma(H)$. If $\lambda = \mu + \imath\nu \in \mathbb{C}$ and $\mu \neq 0$, $\nu \neq 0$, then $-\lambda, \bar{\lambda}, -\bar{\lambda} \in \sigma(H)$.*

It follows that the spectrum of a Hamiltonian matrix can be written as

$$\sigma(H) = \{ \lambda_1, \ldots, \lambda_n, -\lambda_1, \ldots, -\lambda_n \}, \tag{4}$$

where $\mathrm{Re}\,(\lambda_i) \ge 0$, $i = 1, \ldots, n$.

The algorithms implemented here make extensive use of matrices that are both symplectic and orthogonal. Orthogonal symplectic matrices are essentially products of two kinds of elementary orthogonal symplectic matrices: *symplectic Givens* and *symplectic Householder* matrices[2] [28]. These two matrix types can be defined as follows. Let $P = P(v) = I_n - 2\dfrac{vv^T}{v^Tv}$, $v \in \mathbb{R}^n$, be a Householder matrix, then a *symplectic Householder matrix* is given by

$$P_s(v) = \begin{bmatrix} P(v) & 0 \\ 0 & P(v) \end{bmatrix}.$$

A Givens rotation matrix is symplectic, if the rotation is performed in planes k, $n+k$, $1 \le k \le n$. A *symplectic Givens matrix* is thus defined by

$$G_s(k, c, s) = \begin{bmatrix} C & -S \\ S & C \end{bmatrix},$$

[2] Note that this notation is somewhat misleading: whereas a symplectic Givens matrix is also a Givens rotation matrix in the classical sense, a symplectic Householder matrix is not a standard Householder matrix but the direct sum of two $n \times n$ Householder matrices.

where $c, s \in \mathbb{R}$, $c^2 + s^2 = 1$, and $C = I_n + (c-1)e_k e_k^T$, $S = s e_k e_k^T$.

The following proposition shows that the structure of $2n \times 2n$ orthogonal symplectic matrices permits them to be represented as a pair of $n \times n$ matrices. This can cut both the arithmetic costs and storage for accumulating orthogonal symplectic matrices in half.

Proposition 2 *[28]. An orthogonal matrix $U \in \mathbb{R}^{2n\times 2n}$ is also symplectic if and only if it takes the form*

$$U = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$$

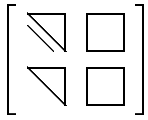
where $U_i \in \mathbb{R}^{n\times n}$, $i = 1, 2$.

2.2 Algorithms

Using the well-known abilities of Householder reflections and Givens rotations to annihilate a specific part of a vector and Proposition 2, the following condensed form of $2n \times 2n$ matrices is derived in [28].

Theorem 3 *If $L \in \mathbb{R}^{2n\times 2n}$, then there exists an orthogonal symplectic matrix $U \in \mathbb{R}^{2n\times 2n}$ such that*

$$\tilde{L} = U^T L U = \begin{bmatrix} \tilde{L}_{11} & \tilde{L}_{12} \\ \tilde{L}_{21} & \tilde{L}_{22} \end{bmatrix} = \qquad (5)$$



i.e., $\tilde{L}_{11}$ is an upper Hessenberg matrix and $\tilde{L}_{21}$ is an upper triangular matrix.

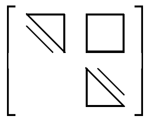
We refer to (5) as *Paige/Van Loan form* or *PVL form.*

Now consider the square of a Hamiltonian matrix H as in (1),

$$K := H^2 = \begin{bmatrix} A^2 + GQ & AG - GA^T \\ QA - A^T Q & (A^2 + GQ)^T \end{bmatrix} =: \begin{bmatrix} K_1 & K_2 \\ K_3 & K_1^T \end{bmatrix}.$$

Since the lower left and the upper right block of K are obviously skew-symmetric, their diagonal is zero. Squared Hamiltonian matrices are *skew-Hamiltonian*, i.e., satisfy $(HJ)^T = -(HJ)^T$. It is easy to see that the structure of skew-Hamiltonian matrices is also preserved by symplectic similarity transformations. Hence, if $S \in \mathbb{R}^{2n\times 2n}$ is a symplectic matrix and H is Hamiltonian, then $S^{-1}H^2 S$ is also a skew-Hamiltonian matrix. Therefore, using Theorem 3, the PVL form of a skew-Hamiltonian matrix $K = H^2$ is

$$\tilde{K} = U^T K U = \begin{bmatrix} \tilde{K_1} & \tilde{K_2} \\ 0 & \tilde{K_1}^T \end{bmatrix} = \qquad (6)$$



The eigenvalues of K can thus be computed by computing the eigenvalues of $\tilde{K_1}$. If $\sigma(\tilde{K_1}) = \{\mu_1, \ldots, \mu_n\}$, then $\sigma(K) = \{\mu_1, \mu_1, \ldots, \mu_n, \mu_n\}$. Hence, according to (4), $\sigma(H)$ can be obtained by taking the positive and negative square roots of the computed eigenvalues of $\tilde{K_1}$. μ_i such that $\lambda_i = \sqrt{\mu_i}$ and $-\lambda_i = -\sqrt{\mu_i}$ for $i = 1, \ldots, n$.

This gives rise to the following algorithm.

Algorithm 4 (Square-reduced method — explicit version)

Input: *A Hamiltonian matrix* $H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$.

Output: $\sigma(H) = \{\lambda_1, \ldots, \lambda_{2n}\}$, *an orthogonal symplectic Matrix* U *such that* $U^T H^2 U$ *has the form (6).*

1. *Compute* $K = H^2 = \begin{bmatrix} A^2 + GQ & AG - GA^T \\ QA - A^T Q & (A^2 + GQ)^T \end{bmatrix}$.

2. *Compute the PVL form of K as described in [28], i.e., determine* $U \in \mathbb{R}^{2n \times 2n}$ *orthogonal and symplectic such that*

$$U^T K U = \begin{bmatrix} \tilde{K}_1 & \tilde{K}_2 \\ 0 & \tilde{K}_1^T \end{bmatrix} = \begin{bmatrix} \diagdown & \square \\ & \diagdown \end{bmatrix}.$$

3. *Compute* $\sigma(\tilde{K}_1) = \{\mu_1, \ldots, \mu_n\}$ *using the Hessenberg QR algorithm.*

4. *Set* $\lambda_i = \sqrt{\mu_i}$, $\lambda_{n+i} = -\sqrt{\mu_i}$ *for* $i = 1, \ldots, n$.

END

If the algorithm is used to approximate the eigenvalues of H only, it is not necessary to accumulate the similarity transformations. This is the case, for instance, in some $\mathcal{H}_\infty$–control applications [20, 9, 8, 10, 38] or the distance-to-instability problem [16, 37].

If, for any reason, the Hamiltonian matrix cannot be overwritten by K, the algorithm requires an additional workspace of size $2n^2 + O(n)$. This is the case when the eigenvalues of the Hamiltonian matrix are used as shifts in CARE solution methods as in [1, 2, 3, 32, 34, 35] since the original Hamiltonian matrix (or a similar Hamiltonian matrix) is needed in following steps of the algorithm. This additional workspace can be avoided using the implicit version of the algorithm as given in [39]. That is, the orthogonal symplectic matrix U from Step 2 of Algorithm 4 is computed without explicitly forming K and is applied to the original Hamiltonian matrix H using

$$U^T K U = (U^T H U)^2.$$

The Hamiltonian matrix $U^T H U$ obtained in this way is said to be in *square-reduced form.* In other words, a square-reduced Hamiltonian matrix H satisfies

$$H^2 = \begin{bmatrix} \diagdown & \square \\ & \diagdown \end{bmatrix}$$

as in (6). Since $U^T H U$ is a Hamiltonian matrix similar to H, the abovementioned algorithms can proceed with $U^T H U$ instead of H and thus need no additional workspace to save the original Hamiltonian matrix. Using this approach, it is necessary to accumulate the orthogonal symplectic similarity transformations. The workspace and computational cost for this can essentially be halved using Proposition 2.

The skew-Hamiltonian Schur form of the skew-Hamiltonian matrix $K = H^2$ used in the CARE solution methods proposed in [42, 43] can be computed by a small modification of

Algorithm 4. Applying the Hessenberg QR algorithm to $\tilde{K}_1$ (Step 3 of Algorithm 4), we obtain an orthogonal matrix $\tilde{V} \in \mathbb{R}^{n\times n}$ such that $T_1 = \tilde{V}^T \tilde{K}_1 V$ is in real Schur form. By Proposition 3, the block diagonal matrix

$$V = \begin{bmatrix} \tilde{V} & 0 \\ 0 & \tilde{V} \end{bmatrix}$$

is orthogonal and symplectic. The *skew-Hamiltonian Schur form* is given by

$$\hat{K} = V^T U^T K U V = \begin{bmatrix} T_1 & T_2 \\ 0 & T_1^T \end{bmatrix}.$$

It is also possible to avoid explicitly squaring the Hamiltonian matrix H in Algorithm 4. This leads to the following implicit version. Note that the same orthogonal symplectic transformations as in Step 2 of Algorithm 4 (computing the PVL form) are computed. Here, they are applied to H rather than to $K = H^2$.

Algorithm 5 (Square-reduced method — implicit version)

Input: *A Hamiltonian matrix* $H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \in \mathbb{R}^{2n\times 2n}$ *and an orthogonal symplectic matrix* $U = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}$.

Output: $\sigma(H)$, *an orthogonal symplectic matrix U such that $U^T H U$ is square-reduced. H is overwritten by its square-reduced form $U^T H U$.*

1. *FOR* $k = 1, \ldots, n-1$

   (a) *IF* $k \le n-2$ *THEN*
   $w \leftarrow (QA - A^T Q)_{k+1:n,k}$
   Compute a Householder reflection P such that $Pw = \pm\|w\|_2 e_1$.
   Update the Hamiltonian matrix via
   $A \leftarrow PAP, \qquad G \leftarrow PGP, \qquad Q \leftarrow PQP.$
   Accumulate the reflection (if required) via
   $U_1 \leftarrow U_1 P, \quad U_2 \leftarrow U_2 P.$
   END IF

   (b) $x \leftarrow (A^2 + GQ)_{k+1,k}, \quad y \leftarrow (QA - A^T Q)_{k+1,k}.$
   Compute a symplectic Givens rotation defined by $c, s \in \mathbb{R}$ such that
   $$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \pm\sqrt{x^2+y^2} \\ 0 \end{bmatrix}$$
   and $C = I_n + (c-1)e_k e_k^T$, $S = s e_k e_k^T$. Update the Hamiltonian matrix via
   $A \leftarrow CAC + SQC + CGS - SA^T S,$
   $G \leftarrow CGC - SA^T S - CAS - SQS,$
   $Q \leftarrow CQC - SAC - SGS - CA^T S.$
   Accumulate the rotation (if required) via
   $U_1 \leftarrow U_1 C + U_2 S, \quad U_2 \leftarrow U_2 C - U_1 S.$

(c) IF $k \leq n-2$ THEN

$w \leftarrow (A^2 + GQ)_{k+1:n,k}$

Compute a Householder reflection P such that $Pw = \pm\|w\|_2 e_1$.

Update the Hamiltonian matrix via

$A \leftarrow PAP, \qquad G \leftarrow PGP, \qquad Q \leftarrow PQP.$

Accumulate the reflection (if required) via

$U_1 \leftarrow U_1 P, \quad U_2 \leftarrow U_2 P.$

END IF

END FOR

2. $W \leftarrow A^2 + GQ$.

3. *Compute $\sigma(W) = \{\mu_1, \ldots, \mu_n\}$ by Hessenberg QR iteration.*

4. $\lambda_i \leftarrow \sqrt{\mu_i}$ *for $i = 1, \ldots, n$ and set $\sigma(H) = \{\lambda_1, \ldots, \lambda_n, -\lambda_1, \ldots, -\lambda_n\}$.*

END

The algorithm as stated above needs the following work space.

	$2n^2 + n$	for the Hamiltonian matrix H,
	n^2	for w and W,
	$2n^2$	for the orthogonal symplectic transformation matrix U.
total	$5n^2 + n$	

The workspace for U is of course not required if only the eigenvalues of the Hamiltonian matrix H are to be computed.

As it is obvious from this formulation of the implicit algorithm, the symplectic Householder and Givens transformation matrices need not be computed explicitly. Updating and accumulating is straightforward and uses well-known techniques, e.g., [23, 40].

Using flop[3] counts and estimates from [23], we get the following estimate for the computational cost of Algorithm 5.

$4n^3$			flops for implicitly computing the columns of H^2,
$3n^2$	+	$O(n)$	flops for generating Householder reflections and Givens rotations,
$16n^3$	+	$O(n^2)$	flops for updating the Hamiltonian matrix,
$8n^3$	+	$O(n^2)$	flops for accumulating the orthogonal symplectic transformations,
$2n^3$	+	$O(n^2)$	flops for forming $W = A^2 + GQ$,
$\frac{20}{3}n^3$	+	$O(n^2)$	flops for the Hamiltonian QR iteration,
n			flops for the square roots in Step 4.
$36\frac{2}{3}n^3$	+	$O(n^2)$	flops total

As for the work space, the computational cost for accumulating the similarity transformations can be saved if only eigenvalues are required. Thus, computing only the eigenvalues of the Hamiltonian matrix requires about $29n^3$ flops which is a little more than one third of the computational cost of about $80n^3$ flops for the QR algorithm as given, e.g., in [23] applied to the $2n \times 2n$ Hamiltonian matrix H. (This flop count includes $80n^3/3$ flops for Hessenberg reduction plus approximately $160n^3/3$ flops for the Hessenberg QR iteration.)

[3]Following [23], we define each floating point arithmetic operation together with the associated integer indexing as a flop.

2.3 Accuracy

Another important issue is the accuracy of the computed eigenvalues. Let $\lambda \in \sigma(H)$ and let $\tilde{\lambda}$ be the analogue computed by either Algorithm 4 or Algorithm 5. Then Van Loan proves in [39] that

$$\tilde{\lambda} \in \sigma\,(H+E),$$

where $E \in \mathbb{R}^{2n\times 2n}$ satisfies

$$\|E\|_2 \leq c\sqrt{\varepsilon}\|H\|_2.$$

Here, c is a small constant and ε denotes the machine precision.

Remark 6 *The perturbation E can be considered as a structured perturbation, that is, if the algorithm is implemented carefully, E is a Hamiltonian matrix.*

The following result for a simple eigenvalue computed by Algorithm 4 or 5 is derived in [39].

$$|\lambda - \tilde{\lambda}| \approx \min\left\{\frac{\varepsilon\|H\|_2^2}{s(\lambda)|\lambda|}, \frac{\sqrt{\varepsilon}\|H\|_2}{s(\lambda)}\right\} = \varepsilon\frac{\|H\|_2}{s(\lambda)} \times \min\left\{\frac{\|H\|_2}{|\lambda|}, \frac{1}{\sqrt{\varepsilon}}\right\} \tag{7}$$

where $s(\lambda)$, the reciprocal condition number of λ, is the cosine of the acute angle between the left and right eigenvectors of H corresponding to λ.

The estimate (7) can be interpreted using the standard estimate for eigenvalues λ^{QR} computed by the Hessenberg QR iteration [40],

$$|\lambda - \lambda^{QR}| \approx \varepsilon\frac{\|H\|_2}{s(\lambda)},$$

as follows:

- If $\|H\|_2/\lambda \approx 1$, then $\tilde{\lambda}$ is as accurate as λ^{QR},
- If $\|H\|_2/|\lambda| \gg 1$, then the error in $\tilde{\lambda}$ can be up to $1/\sqrt{\varepsilon}$ times larger than the error in λ^{QR}.

Thus, if relatively small eigenvalues of a Hamiltonian matrix are required with high accuracy, it is not advisable to use Van Loan's method to compute the eigenvalues. However, if the approximate eigenvalues are used, for example, as shifts, the highest possible accuracy is not required. In the multishift and related algorithms [1, 2, 3, 34], more accurate eigenvalues can be obtained during the iteration. Sometimes, accurate shifts may even lead to forward instability, see [30]. For a discussion of the problem of eigenvalues close to the imaginary axis (including "small" eigenvalues) see Section 3.2.

3 Implementation

For our implementation, we chose the implicit version of the algorithm in order to avoid the need of explicitly computing the square of the Hamiltonian matrix. Moreover, the Hamiltonian matrix is transformed to a similar Hamiltonian matrix which can then further be used, e.g., in CARE solution algorithms. This avoids saving a copy of the original Hamiltonian matrix. The subroutine DHASRD implements the first step of Algorithm 5, i.e., computes the square-reduced Hamiltonian and, if required, the orthogonal symplectic transformation matrix. The subroutine DHAEVS performs steps 2–4 of Algorithm 5 using the output from DHASRD. We chose to separate these steps in order to make it possible to use the square-reduced Hamiltonian matrix itself. We also supply an easy-to-use driver DHAEVD that calculates the eigenvalues of a Hamiltonian matrix using DHASRD and DHAEVS. The implementation and documentation of all subroutines follow the standards proposed for the **S**ubroutine **L**ibrary **i**n **Co**ntrol **T**heory SLICOT (see [41]). This standard is based on the documentation style of LAPACK [4].

We will now describe some implementation details and how the subroutines are used. The subroutines require DOUBLE PRECISION versions of the BLAS (levels 1,2,3) [27, 19, 18] and LAPACK [4]. With $fl(\circ)$ we will denote the result of the floating point operation "$\circ$" in finite arithmetic.

3.1 Scaling

It is sometimes possible to use scaling or balancing to improve the accuracy of computed eigenvalues. There are several possibilities for scaling in Algorithms 4 and 5. Since *balancing* as proposed in [31] destroys the Hamiltonian structure, we have to look for balancing strategies involving only symplectic similarity transformations. Balancing via diagonal matrices and preserving the Hamiltonian structure leads necessarily to matrices of the form

$$D_s = \begin{bmatrix} D & 0 \\ 0 & D^{-1} \end{bmatrix} \tag{8}$$

where D is diagonal. We propose three simple, semi-heuristic strategies for choosing the diagonal scaling matrix D, *symplectic scaling*, *norm scaling*, and *Hessenberg scaling*.

3.1.1 Symplectic Scaling

Symplectic scaling is a two part strategy. First a diagonal matrix $D = D_A$ is chosen to to equilibrate the norms of the rows and columns of $D_A^{-1}AD_A$ [31]. This transformation can be computed, e.g., by the LAPACK [4] subroutine DGEBAL. This similarity transformation of A is extended to a similarity transformation on the Hamiltonian matrix H by a symplectic, diagonal matrix D_s as in (8). The effect of this is that the rows and columns of $\tilde{A} := D_A^{-1}AD_A$ have approximately the same 1-norm which often has also the effect of reducing the 1-norm of A. Thus, if this diagonal scaling yields a reduction of the norm of A (say, by a factor $\gamma < 1$), we can hope to improve the accuracy for $fl(\tilde{A}^2)$ since ([23])

$$\|fl(\tilde{A}^2) - \tilde{A}^2\|_1 \leq n\varepsilon\|\tilde{A}\|_1^2 + O(\varepsilon^2) = n\varepsilon\gamma\|A\|_1^2 + O(\varepsilon^2) < n\varepsilon\|A\|_1^2 + O(\varepsilon^2).$$

The second part of the symplectic scaling strategy is a scaling strategy used for improving the accuracy of solution methods for the CARE (2). Here, the diagonal matrix D in (8) is chosen as $D_\rho = I_n/\sqrt{\rho}$. This scaling strategy results in a Hamiltonian matrix

$$\tilde{H} := \begin{bmatrix} A & \tilde{G} \\ \tilde{Q} & -A^T \end{bmatrix} := \begin{bmatrix} {D_\rho}^{-1} & 0 \\ 0 & D_\rho \end{bmatrix} \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \begin{bmatrix} D_\rho & 0 \\ 0 & {D_\rho}^{-1} \end{bmatrix} \tag{9}$$

where $\tilde{G} = \rho G$ and $\tilde{Q} = Q/\rho$. It was observed in [33] that scaling H such that $\|A\| = \|G\| = \|Q\|$ is optimal in the sense that it minimizes the error bounds for the numerical solution of the CARE (2). Unfortunately, it is in general not possible to achieve this optimal scaling via (9). Noting

$$\|\tilde{H}\|_1 \leq \|A\|_1 + \max\{\|\tilde{G}\|_1, \|\tilde{Q}\|_1\} \tag{10}$$

and recalling that the error of the computed eigenvalues depends upon $\|H\|^2$ we choose ρ to minimize the bound in (10). This is achieved by $\rho = \sqrt{\|Q\|_1/\|G\|_1}$, such that $\|\tilde{G}\|_1 = \|\tilde{Q}\|_1$ [22].

Both ideas can be combined since the second scaling strategy will not affect $D_A^{-1}AD_A$. We will refer to this two-step scaling as symplectic scaling since the scaling is performed by a similarity transformation with a symplectic matrix. The resulting Hamiltonian matrix is

$$\tilde{H} = \begin{bmatrix} D_A^{-1}AD_A & \rho D_A^{-1}GD_A^{-1} \\ \frac{1}{\rho}D_AQD_A & -D_AA^TD_A^{-1} \end{bmatrix}$$

where $\rho = \sqrt{\|D_AQD_A\|_1/\|D_A^{-1}GD_A^{-1}\|_1}$.

3.1.2 Norm Scaling

Norm scaling is the strategy proposed in [24, 29] for stabilizing the Schur vector method for the solution of CAREs (2). Consider the CARE

$$0 = Q + (\frac{1}{\tau}A)^T(\tau X) + (\tau X)(\frac{1}{\tau}A) - (\tau X)(\frac{1}{\tau^2}G)(\tau X).$$

The corresponding Hamiltonian matrix is

$$\hat{H} = \begin{bmatrix} \hat{A} & \hat{G} \\ Q & -\hat{A}^T \end{bmatrix}$$

where $\hat{A} = A/\tau$, $\hat{G} = G/\tau^2$, and $\lambda \in \sigma(H)$ if and only if $\hat{\lambda} = \lambda/\tau \in \sigma(\hat{H})$. This strategy is equivalent to scaling the Hamiltonian matrix H by the real scalar $1/\tau$ and then performing a symplectic similarity transformation with D_s from (8) where $D = \sqrt{\tau}I_n$. Using $\hat{H}$ instead of H in Algorithm 5, we obtain ([23])

$$\begin{aligned} \|fl(\hat{A}^2 + \hat{G}Q) - (\hat{A}^2 + \hat{G}Q)\|_1 &\leq n\varepsilon\frac{1}{\tau^2}(\|A\|_1^2 + \|G\|_1\|Q\|_1) + O(\varepsilon^2), \qquad (11) \\ \|fl(Q\hat{A} - \hat{A}^TQ) - (Q\hat{A} - \hat{A}^TQ)\|_1 &\leq 2n\varepsilon\frac{1}{\tau}\|A\|_1\|Q\|_1 + O(\varepsilon^2). \qquad (12) \end{aligned}$$

The choice $\tau = \max\{\|A\|_1, \|G\|_1, \|Q\|_1\}$ reduces the absolute errors given above to $O(n\varepsilon)$ and $O(n\varepsilon\|H\|_1)$, respectively. We bound τ by one from below in order not to increase the absolute errors (11) and (12).

3.1.3 Hessenberg Scaling

In Step 3 of Algorithm 5, the usual balancing as proposed in [31] of a matrix can be applied to W using LAPACK subroutine DGEBAL. This type of scaling will be called *Hessenberg scaling* since it is applied to the upper left Hessenberg block of the squared Hamiltonian matrix.

3.2 Eigenvalues on or close to the imaginary axis

An important issue in many applications of the Hamiltonian eigenproblem are eigenvalues on or close to the imaginary axis. Algorithms for the computation of the $\mathcal{H}_\infty$–norm of transfer matrices [9, 8, 20, 10, 38] or the stability radius of real and complex matrices [16, 37] have to decide whether there are eigenvalues on the imaginary axis or not. Knowledge of the spectrum of H also gives information about the solution of the CARE (2).

Theorem 7 *Let H be a Hamiltonian matrix as in (1) and assume there exist full-rank factorizations $G = BB^T$, $Q = C^TC$. If (A, B) is stabilizable, i.e., rank$[A - \lambda I_n,\ B] = n$ for all $\lambda \in \mathbb{C}^+ \cup \imath\mathbb{R}$, and (A, C) is detectable, i.e, (A^T, C^T) stabilizable, then*
a) $Re(\lambda) \neq 0$ for all $\lambda \in \sigma(H)$.
b) The CARE (2) has a unique symmetric positive semidefinite stabilizing solution $\hat{X}$.
c) If $\sigma(H)$ is as in (4), then $\sigma(A - G\hat{X}) = \{\, -\lambda_1, \ldots, -\lambda_n \,\}$.

Clearly, if the Hamiltonian matrix corresponding to the CARE has eigenvalues on the imaginary axis, then no stabilizing solution can exist. On the other hand, if all Jordan blocks corresponding to eigenvalues on the imaginary axis have even size, then a real n–dimensional H–invariant *Lagrangian*[4] subspace corresponding to eigenvalues λ, $\mathrm{Re}(\lambda) \leq 0$, exists and under additional assumptions, an *almost stabilizing* solution $\tilde{X}$ of the CARE can be computed, that is, $\sigma(A - G\tilde{X}) \subset \mathbb{C}^- \cup \imath\mathbb{R}$. For a discussion of the relation of the spectra of Hamiltonian matrices to the existence and uniqueness of solutions of the CARE we refer to [25]. CARE solution methods that require an a priori knowledge of the spectrum of H can therefore use this information to decide if a solution of (2) exists before trying to compute it. This is the case, e.g., for the multishift and related algorithms [2, 3, 34, 32].

DHABL/DHASRD/DHAEVS is well suited to testing whether a Hamiltonian matrix has eigenvalues on the imaginary axis. Because the computed eigenvalues are eigenvalues of a perturbed Hamiltonian matrix, the computed eigenvalues appear in $\pm$ pairs as in (4). In particular, imaginary axis eigenvalues that are simple and sufficiently isolated will remain on the imaginary axis despite rounding errors. However, multiple or nearly multiple imaginary axis eigenvalues are possible and typical in stability radius and H_∞ norm calculations. Rounding errors can perturb multiple and near multiple eigenvalues off of the imaginary axis. A typical test would be to consider a computed eigenvalue $\tilde{\lambda}$ to have zero real part if

$$|\mathrm{Re}(\tilde{\lambda})| \leq \mathrm{tol} \times |\tilde{\lambda}|$$

where tol is a user-defined tolerance. A conservative choice of tol is $\sqrt{\varepsilon}$ where ε is the unit round. This tolerance is motivated by the following fact: suppose we have computed an eigenvalue of H^2 of magnitude $\varepsilon\|H\|^2$. Since rounding errors can perturb eigenvalues by at least $\varepsilon\|H\|^2$, it is problematical to determine whether or not the corresponding exact, unperturbed eigenvalue lies on the negative real axis. Taking its square root can result in a computed eigenvalue of H with real part of size up to $\sqrt{\varepsilon}\|H\|$. Conversely, if the Hamiltonian matrix has an exact eigenvalue with magnitude $O(\sqrt{\varepsilon})$, then the corresponding eigenvalue of H^2 has magnitude $O(\varepsilon)$. Rounding errors could perturb this small eigenvalue of H^2 onto the negative real axis making it appear as if H had an eigenvalue on the imaginary axis.

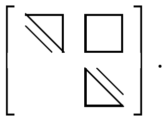
In Section 5 we will give an example how this can be used when computing the stability radius of a real matrix or the $\mathcal{H}_\infty$–norm of a transfer matrix.

[4] An n-dimensional subspace $\mathcal{V} \subset \mathbb{R}^{2n}$ is called *Lagrangian* if $x^TJy = 0$ for all $x, y \in \mathcal{V}$ and J as in (3).

3.3 Usage

The subroutine DHASRD computes the square-reduced form of a Hamiltonian matrix, i.e., if $\tilde{H}$ is the Hamiltonian matrix returned by DHASRD, then (in exact arithmetic)

$$\tilde{H}^2 = \begin{bmatrix} K_1 & K_2 \\ 0 & K_1^T \end{bmatrix} =$$



The subroutine is called as follows:

```
CALL DHASRD(COMPU, N, A, LDA, QG, LDQG, U, LDU, RWORK, IERR)
```

A complete description of input and output arguments is given in the prologue in the DHASRD source code. The Hamiltonian matrix (1) is provided by its blocks A, G, and Q. The block A is stored in the two dimensional array `A`. The lower triangular part of the symmetric matrix Q and the upper triangular part of the symmetric matrix G are packed together in the $n \times (n+1)$ array `QG`.

If required (as defined by `COMPU`), the orthogonal symplectic transformations are accumulated in a matrix U. The output argument `U` is a two dimensional array that contains the first n rows of this matrix. By Proposition 2, the first n rows completely define the $2n \times 2n$ orthogonal symplectic matrix U.

The subroutine DHAEVS uses the output of DHASRD and computes the eigenvalues of the Hamiltonian matrix. The calling sequence is

```
CALL DHAEVS(JOBSCL, N, A, LDA, QG, LDQG, WR, WI, RWORK, IERR)
```

A complete description of input and output arguments is given in the prologue in the DHAEVS source code. Briefly, `A` and `QG` are the arrays computed by DHASRD. The eigenvalues of the Hamiltonian matrix are returned in `WR` (real parts) and `WI` (imaginary parts). If the mode parameter `JOBSCL` is set to `S`, then DHAEVS will do Hessenberg scaling as described in Section 3.1.

The subroutine DHAEVD is an easy-to-use driver that calls DHASRD and DHAEVS to calculate the eigenvalues of a Hamiltonian matrix. The calling sequence is

```
CALL DHAEVD(N, A, LDA, QG, LDQG, WR, WI, RWORK, IERR)
```

A complete description of input and output arguments is given in the prologue in the source code of DHAEVD . To make this program as easy to use as possible, there are no mode parameters. DHAEVD uses Hessenberg scaling. The symplectic orthogonal similarity transformation is not accumulated. As with DHASRD and DHABL, the Hamiltonian matrix (1) is provided by its blocks A, G, and Q. The block A is stored in the two dimensional array `A`. The lower triangular part of the symmetric matrix Q and the upper triangular part of the symmetric matrix G are packed together in the $n \times (n+1)$ array `QG`.

The symplectic and norm scaling strategies are implemented in subroutine DHABL with calling sequence

```
CALL DHABL(JOBSCL, N, A, LDA, QG, LDQG, D, RWORK, IERR)
```

A complete description of input and output arguments is given in the prologue in the DHABL source code. When the mode parameter `JOBSCL` is set to "`A`", DHABL performs symplectic scaling. When the mode parameter `JOBSCL` is set to "`B`", DHABL performs norm scaling. (Hessenberg scaling is performed by subroutine DHAEVS.)

SUBROUTINES CALLED BY DHAEVS:	
BLAS:	DCOPY, DGEMM, DSYMV.
LAPACK:	DGEBAL, DHSEQR, DLACPY, LSAME.
Miscellaneous:	DCROOT (which calls LAPACK DLAPY2)
SUBROUTINES CALLED BY DHASRD:	
BLAS:	DAXPY, DCOPY, DDOT, DGEMV, DROT, DSYMV, DSYR2.
LAPACK:	DLARFG, DLARFX, DLARTG, LSAME.
SUBROUTINES CALLED BY DHABL:	
BLAS:	DRSCL.
LAPACK:	DGEBAL, DLABAD, DLAMCH, DLANGE, DLANSY, DLASCL, LSAME.

Table 1: Required BLAS and LAPACK subroutines.

3.4 Subroutine organization

The basis for the subroutines DHASRD and DHAEVS are the BLAS and LAPACK [27, 19, 4]. Table 1 shows the functions and subroutines required from these subroutine libraries.

The easy-to-use driver DHAEVD calls DHASRD and DHAEVS. Besides the BLAS and LAPACK routines, DHAEVS calls a short routine named DCROOT to compute the square root of a complex number using real arithmetic. The root is chosen such that the real part is nonnegative. This subroutine was adapted from the EISPACK [36] subroutine CSROOT.

4 Numerical Examples

We tested our subroutines for all Hamiltonian matrices from the benchmark collection for continuous-time algebraic Riccati equations [6], the examples given in [39], and some randomly generated examples. Here, we present some selected experiments.

The numerical tests reported here used IEEE double precision arithmetic with machine precision $\varepsilon \approx 2.2204 \times 10^{-16}$. The programs were compiled and run on a Sun Microsystems Ultra-1-200E computer, using version 4.2 of the Sun f77 compiler with the `-dalign -fsimple=1 -libmil -xlic_lib=sunperf` compiler flags. The last flag instructs the compiler to use LAPACK subroutines and tuned BLAS from the Sun performance library version 1.2.

We also run several tests on Hewlett-Packard workstations and PCs running LINUX. The effects of rounding are similar to those reported below. Variations in execution time between the various platforms appear to be due to differing CPUs.

In the following tests, when exact eigenvalues are unavailable, we will assess the computed eigenvalues in the following backward error sense. Suppose $\tilde{\lambda}$ is an approximate eigenvalue of

DHABL(A)/DHASRD/DHAEVS(S)	Symplectic Scaling and Hessenberg scaling
DHABL(B)/DHASRD/DHAEVS(S)	Norm Scaling and Hessenberg scaling
DHASRD/DHAEVS(S)	Hessenberg scaling alone
DHASRD/DHAEVS	No scaling

Table 2: Notation for the scaling strategies discussed in Section 3.1.

a matrix H. Define $\sigma_{\tilde{\lambda}}$ to be the smallest singular value of $H - \tilde{\lambda}I$, i.e., $\sigma_{\tilde{\lambda}} = \sigma_{\min}(H - \tilde{\lambda}I)$. It is well known that $\sigma_{\tilde{\lambda}}$ is the 2-norm of the perturbation matrix E of smallest 2-norm for which $\tilde{\lambda}$ is an exact eigenvalue of $H + E$. In other words,

$$\sigma_{\tilde{\lambda}} = \min\left\{ \|E\| \;\middle|\; \tilde{\lambda} \in \sigma(H - \tilde{\lambda}I) \right\}.$$

Thus, $\sigma_{\tilde{\lambda}}$ is the magnitude of the "backward error" in $\tilde{\lambda}$ as an approximate eigenvalue of H. In particular, the smaller $\sigma_{\tilde{\lambda}}$, the better is $\tilde{\lambda}$ as an approximate eigenvalue of H.

In this section we will denote the various scaling strategies discussed in Section 3.1 as shown in Table 2.

We compared our subroutines with the LAPACK driver routine DGEEVX for computing the eigenvalues of a general real matrix using the QR algorithm. In all calls to DGEEVX, we used (non-symplectic) diagonal scaling and permutations (BALANC='B') to improve the conditioning of eigenvalues. Although rounding errors prevent LAPACK subroutine DGEEVX from preserving the Hamiltonian eigenvalue plus-minus pairs, the tables below report only the computed eigenvalues with non-negative real part.

Example 1 [39, Example 2] Let

$$A = \mathrm{diag}(1,\ 10^{-2},\ 10^{-4},\ 10^{-6},\ 10^{-8}).$$

Then a Hamiltonian matrix H is obtained by

$$H = U^T \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} U,$$

where $U \in \mathbb{R}^{2n\times 2n}$ is an orthogonal symplectic matrix randomly generated by five symplectic rotations and five reflectors. Thus, $\sigma(H) = \{\pm 1, \pm 10^{-2}, \pm 10^{-4}, \pm 10^{-6}, \pm 10^{-8}\}$.

For this example all the selected scale parameters equal one and hence, all the scaling strategies produced the same results. Table 3 shows the absolute errors in the eigenvalue approximations computed by DHASRD/DHAEVS and DGEEVX. Here, the loss of accuracy of order $\|H\|_2/|\lambda|$ for Van Loan's method is obvious. DGEEVX computes all eigenvalues to full accuracy. Repeating the computations for several randomly chosen matrices U, the results for DHASRD/DHAEVS varied by about one order of magnitude for the smaller eigenvalues. (This is the reason for the slightly different results reported in [7].)

λ	DHASRD DHAEVS	DGEEVX
1	0	$1.0 \cdot 10^{-15}$
10^{-2}	$5.0 \cdot 10^{-15}$	$5.0 \cdot 10^{-17}$
10^{-4}	$3.5 \cdot 10^{-14}$	$2.6 \cdot 10^{-18}$
10^{-6}	$1.4 \cdot 10^{-11}$	$8.4 \cdot 10^{-18}$
10^{-8}	$2.0 \cdot 10^{-9}$	$4.7 \cdot 10^{-17}$

Table 3: Example 1, absolute errors $|\lambda - \tilde{\lambda}|$

Example 2 [39, Example 3] The *Frank* matrix $A \in \mathbb{R}^{n\times n}$ is defined by

$$A = \begin{bmatrix} n & n-1 & n-2 & \dots & \dots & 2 & 1 \\ n-1 & n-1 & n-2 & \dots & \dots & 2 & 1 \\ 0 & n-2 & n-2 & \dots & \dots & 2 & 1 \\ 0 & 0 & n-3 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & & \ddots & 2 & 1 \\ 0 & 0 & \dots & & 0 & 1 & 1 \end{bmatrix}.$$

All its eigenvalues are real and positive. The smaller eigenvalues are ill-conditioned. A Hamiltonian matrix having the same eigenvalues as the Frank matrix together with their negative counterparts is generated as in Example 1 by

$$H = U^T \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} U,$$

where $U \in \mathbb{R}^{2n\times 2n}$ is an orthogonal symplectic matrix randomly generated by n symplectic rotations and n reflectors.

Exact eigenvalues are not known and some eigenvalues are ill-conditioned, so Table 4 lists the backward errors $\sigma_{\tilde{\lambda}} = \sigma_{\min}(H - \tilde{\lambda} I)$. The table demonstrates that, at least in this case, norm scaling DHABL(B)/DHASRD/DHAEVS(S) is not as effective as the other scaling strategies.

Example 3 [6, Example 11] Let

$$A = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix}, \qquad G = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad Q = \begin{bmatrix} -11 & -5 \\ -5 & -2 \end{bmatrix}.$$

The spectrum of $H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix}$ is $\{\pm\imath\}$ with algebraic multiplicity two and geometric multiplicity one. The computed eigenvalues are listed in Table 5.

$\lambda \approx$	DHABL(A) DHASRD DHAEVS(S)	DHABL(B) DHASRD DHAEVS(S)	DHASRD DHAEVS(S)	DHASRD DHAEVS	DGEEVX
.031028	$9.1 \cdot 10^{-14}$	$5.5 \cdot 10^{-11}$	$2.8 \cdot 10^{-14}$	$6.1 \cdot 10^{-14}$	$1.8 \cdot 10^{-15}$
.049508	$5.2 \cdot 10^{-14}$	$3.5 \cdot 10^{-11}$	$1.8 \cdot 10^{-14}$	$3.3 \cdot 10^{-14}$	$2.1 \cdot 10^{-15}$
.081228	$2.6 \cdot 10^{-14}$	$1.9 \cdot 10^{-11}$	$7.9 \cdot 10^{-15}$	$1.4 \cdot 10^{-14}$	$1.8 \cdot 10^{-15}$
.14365	$6.0 \cdot 10^{-15}$	$9.6 \cdot 10^{-12}$	$1.4 \cdot 10^{-15}$	$1.9 \cdot 10^{-15}$	$1.6 \cdot 10^{-15}$
.28475	$7.4 \cdot 10^{-15}$	$3.6 \cdot 10^{-12}$	$1.0 \cdot 10^{-14}$	$7.9 \cdot 10^{-15}$	$1.7 \cdot 10^{-15}$
.64351	$1.5 \cdot 10^{-14}$	$8.9 \cdot 10^{-13}$	$1.5 \cdot 10^{-14}$	$1.4 \cdot 10^{-14}$	$1.5 \cdot 10^{-15}$
1.5540	$5.2 \cdot 10^{-15}$	$1.3 \cdot 10^{-13}$	$7.5 \cdot 10^{-15}$	$7.7 \cdot 10^{-15}$	$6.5 \cdot 10^{-16}$
3.5119	$1.6 \cdot 10^{-15}$	$3.8 \cdot 10^{-14}$	$3.7 \cdot 10^{-16}$	$3.7 \cdot 10^{-16}$	$3.2 \cdot 10^{-15}$
6.9615	$5.0 \cdot 10^{-16}$	$4.1 \cdot 10^{-14}$	$5.3 \cdot 10^{-15}$	$1.3 \cdot 10^{-15}$	$7.2 \cdot 10^{-15}$
12.311	$7.7 \cdot 10^{-15}$	$2.0 \cdot 10^{-14}$	$3.1 \cdot 10^{-15}$	$4.7 \cdot 10^{-15}$	$1.1 \cdot 10^{-14}$
20.199	$7.0 \cdot 10^{-15}$	$5.5 \cdot 10^{-16}$	$3.6 \cdot 10^{-15}$	$6.0 \cdot 10^{-15}$	$6.8 \cdot 10^{-15}$
32.229	$1.0 \cdot 10^{-14}$	$1.5 \cdot 10^{-14}$	$4.9 \cdot 10^{-15}$	$2.8 \cdot 10^{-15}$	$2.0 \cdot 10^{-15}$

Table 4: Example 2, backward errors $\sigma_{\tilde{\lambda}} = \sigma_{\min}(H - \tilde{\lambda} I)$ for computed eigenvalues $\tilde{\lambda}$.

Subroutines	**Computed Eigenvalues**
DHABL(A) DHASRD DHAEVS(S)	$0.187133296480218 \cdot 10^{-7} \pm 1.000000000000000i$ $-0.187133296480218 \cdot 10^{-7} \pm 1.000000000000000i$
DHABL(B) DHASRD DHAEVS(S)	$\pm 0.999999971617234i$ $\pm 1.000000028382765i$
DHASRD DHAEVS(S)	$\pm 0.999999968884276i$ $\pm 1.000000031115725i$
DHASRD DHAEVS	$\pm 0.999999968884276i$ $\pm 1.000000031115725i$
DGEEVX	$0.589883505930366 \cdot 10^{-8} \pm 1.000000028925357i$ $-0.589883537155389 \cdot 10^{-8} \pm 0.999999971074641i$

Table 5: Example 3, computed eigenvalues.

$\lambda =$	DHABL(A) DHASRD DHAEVS(S)	DHABL(B) DHASRD DHAEVS(S)	DHASRD DHAEVS(S)	DHASRD DHAEVS	DGEEVX
33.3	$1.7 \cdot 10^{-10}$	$1.2 \cdot 10^{-9}$	$3.3 \cdot 10^{-12}$	$2.8 \cdot 10^{-12}$	0
20.0	$2.2 \cdot 10^{-10}$	$4.3 \cdot 10^{-12}$	$5.0 \cdot 10^{-12}$	$9.1 \cdot 10^{-13}$	0
20.0	$1.6 \cdot 10^{-10}$	$2.6 \cdot 10^{-13}$	$3.5 \cdot 10^{-10}$	$3.5 \cdot 10^{-10}$	0
20.0	$2.7 \cdot 10^{-10}$	$2.6 \cdot 10^{-13}$	$3.5 \cdot 10^{-10}$	$3.5 \cdot 10^{-10}$	0

Table 6: Example 4, absolute errors for exact eigenvalues 33.3, 20, 20, 20.

The different scaling strategies for DHAEVS result in different perturbations of the eigenvalues. The QR algorithm as implemented in DGEEVX does not preserve Hamiltonian structure. Its computed eigenvalues do not appear in plus-minus pairs as the exact eigenvalues do. Eigenvalues computed through the square-reduced method remain in plus-minus pairs.

Example 4 [6, Example 6], [17] This example comes from a control problem for a J-100 jet engine as special case of a multivariable servomechanism problem. The corresponding linear time-invariant system is described by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \quad \text{for } t > 0, \qquad x(0) = x_0, \\ y(t) &= Cx(t) \quad \text{for } t \geq 0, \end{aligned}$$

where the state vector x contains the state of the jet engine, the actuators, and the sensors. The system matrices $A \in \mathbb{R}^{30\times 30}$, $B \in \mathbb{R}^{30\times 3}$, and $C \in \mathbb{R}^{5\times 30}$ appear in [6, 17]. The Hamiltonian matrix is

$$H = \begin{bmatrix} A & BB^T \\ C^T C & -A^T \end{bmatrix}.$$

We know the exact values of only four of the eigenvalues of H: 33.3 and a triple eigenvalue at 20.0. DGEEVX computed these four eigenvalues without error. (It returned 33.3 correctly rounded to the precision of the arithmetic.) Table 6 lists absolute errors for these four known eigenvalues. The table shows that norm scaling improved the accuracy of the triple eigenvalue 20.

Table 7 lists the values of $\sigma_{\tilde{\lambda}}$ for the two largest magnitude and two smallest magnitude computed eigenvalues. In this example the norm of the Hamiltonian matrix is relatively large compared to the eigenvalues. The largest magnitude eigenvalue is approximately 577 but $\|H\| \approx 10^8$. The table clearly demonstrates the $O(\sqrt{\varepsilon})$ backward error in the smallest eigenvalues computed through DHASRD.

In this example, norm scaling significantly increased the accuracy of the eigenvalue pair $\pm\lambda$ of smallest magnitude, but at the same time, it also significantly decreased the accuracy of the eigenvalue pair of second smallest magnitude.

Example 5 [26, Example 5], [6, Example 15], [39, Example 1] The Hamiltonian matrix in this problem comes from the position and velocity control problem for a string of N high-speed

$\lambda \approx$	DHABL(A) DHASRD DHAEVS(S)	DHABL(B) DHASRD DHAEVS(S)	DHASRD DHAEVS(S)	DHASRD DHAEVS	DGEEVX
.18240	$4.6 \cdot 10^{-12}$	$1.8 \cdot 10^{-15}$	$1.5 \cdot 10^{-10}$	$1.5 \cdot 10^{-10}$	$6.2 \cdot 10^{-17}$
.64303	$6.1 \cdot 10^{-8}$	$3.4 \cdot 10^{-4}$	$6.6 \cdot 10^{-10}$	$6.6 \cdot 10^{-10}$	$2.5 \cdot 10^{-15}$
187.64	0	$1.9 \cdot 10^{-13}$	$2.0 \cdot 10^{-13}$	$2.2 \cdot 10^{-13}$	$1.6 \cdot 10^{-14}$
577.04	$8.3 \cdot 10^{-14}$	$6.4 \cdot 10^{-14}$	$1.3 \cdot 10^{-12}$	$1.5 \cdot 10^{-12}$	$2.6 \cdot 10^{-14}$

Table 7: Example 4, backward errors $\sigma_{\tilde{\lambda}} = \sigma_{\min}(H - \tilde{\lambda} I)$ for computed eigenvalues $\tilde{\lambda}$.

vehicles and is given by

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 & \dots & & 0 \\ 0 & A_{22} & A_{23} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & 0 & A_{N-2,N-2} & A_{N-2,N-1} & 0 \\ & & & 0 & A_{N-1,N-1} & \begin{bmatrix} 0 \\ -1 \end{bmatrix} \\ 0 & \dots & & & 0 \quad 0 & -1 \end{bmatrix}$$

where

$$A_{k,k} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}, \quad 1 \le k \le N-1, \qquad \text{and} \qquad A_{k,k+1} = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}, \quad 1 \le k \le N-2.$$

The off-diagonal blocks are given by

$$\begin{aligned} G &= \operatorname{diag}(1,\ 0,\ 1,\ 0,\ \dots, 1,\ 0,\ 1), \\ Q &= \operatorname{diag}(0,\ 10,\ 0,\ 10, \dots, 0,\ 10,\ 0). \end{aligned}$$

The Hamiltonian matrix is

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \in \mathbb{R}^{2n \times 2n}$$

with $n = 2N - 1$. All the eigenvalues are relatively well conditioned, and $\|H\|_2/|\lambda| = O(1)$ for all $\lambda \in \sigma(H)$.

In Table 8 we give the CPU times used by DHABL/DHASRD/DHAEVS and by DGEEVX on the SUN Ultra-1-200E. The table shows that for increasing N, the CPU time used by DHAEVS tends to be very roughly 20% of that of DGEEVX. This is faster than the 36% predicted by flop counts and seems to be due to the handling of multiplications with zeros in the SUN CPU. Execution times on a LINUX–PC with an AMD K6 200 MHz CPU are in accordance with the flop counts.

Example 6 We tested our subroutines for randomly generated Hamiltonian matrices with entries distributed uniformly in the interval $[-1, 1]$. Since the eigenvalue distribution for these examples usually behaves nicely, the eigenvalues computed by DHASRD/DHAEVS are as accurate as for DGEEVX even without scaling. We give the CPU times for $2n \times 2n$

$N \quad (n = 2N-1)$	25	50	75	100	200
DHABL(A)/DHASRD/DHAEVS(S)	0.045	0.29	0.90	2.0	17
DHABL(B)/DHASRD/DHAEVS(S)	0.043	0.29	0.90	2.0	17
DHASRD/DHAEVS(S)	0.042	0.28	0.88	2.0	17
DHASRD/DHAEVS	0.042	0.28	0.88	1.9	17
DGEEVX	0.19	1.2	4.4	10	84

Table 8: Example 5, CPU seconds of execution time for DHABL/DHASRD/DHAEVS and DGEEVX.

n	25	50	75	100	125	150	175	200
DHASRD/DHAEVS	0.014	0.083	0.26	0.59	1.1	2.0	3.1	4.6
DGEEVX	0.032	0.20	0.67	1.4	2.8	4.8	7.7	11.

Table 9: Example 6, average CPU seconds of execution times for DHASRD/DHAEVS (without scaling) and DGEEVX across 100 trials.

examples for n varying from 25 to 200. For each value of n, we recorded the CPU seconds of execution time required to solve 100 randomly generated Hamiltonian eigenvalue problems. Table 9 lists the mean CPU seconds of execution time measured on the SUN Ultra-1-200E. The table shows that for these Hamiltonian matrices, DHASRD/DHAEVS uses about 41% of the execution time of DGEEVX. This is slightly more that the 36% predicted by flop counts. This is probably because our implementation of DHASRD does not take as much advantage of the level 3 BLAS as does DGEEVX.

Besides the faster computation of the eigenvalues, DHAEVS returns the right pairing of the eigenvalues as $\pm\lambda_i$, $i = 1, \ldots, n$. Since DGEEVX treats a Hamiltonian matrix like an arbitrary nonsymmetric matrix, small perturbations can and do cause computed eigenvalues with small real parts to cross the imaginary axis. For example, the number of stable eigenvalues returned by DGEEVX for the $n = 100$ case varied from a minimum of 94 to a maximum of 104. Similar variations were observed for all values of n.

5 Applications

In this section, we demonstrate the application of of DHABL/DHASRD/DHAEVS to two problems in computational control theory: calculation of the stability radius and calculation of the H_∞ norm of a transfer function.

Example 7 *The stability radius of a matrix* [16].

Given a stable matrix $A \in \mathbb{R}^{n\times n}$ (i.e., $\sigma(A) \subset \mathbb{C}^-$), it is often important to know how near A is to an unstable matrix, i.e., what is the smallest norm perturbation $E \in \mathbb{C}^{n\times n}$ for which $A + E$ is unstable. The distance of A to the unstable matrices can be measured by

$$\beta(A) = \min\{\|E\| \,:\, \sigma(A+E) \cap \imath\mathbb{R} \neq \emptyset\}.$$

In [16], a bisection method for measuring $\beta(A)$ is presented which is based on the following observation:

Theorem 8 *[16] If $\alpha \geq 0$, then the Hamiltonian matrix*

$$H = H(\alpha) = \begin{bmatrix} A & -\alpha I_n \\ \alpha I_n & -A^T \end{bmatrix}$$

has an eigenvalue on the imaginary axis if and only if $\alpha \geq \beta(A)$.

The following algorithm estimates $\beta(A)$ within a factor of 10 or indicates that $\beta(A)$ is less than a small tolerance.

Algorithm BISEC

Input: $A \in \mathbb{R}^{n\times n}$ and a tolerance $tol > 0$.

Output: $\delta, \gamma \in \mathbb{R}$ satisfying either $\frac{\gamma}{10} \leq \delta \leq \beta(A) \leq \gamma$ or $0 \leq \beta(A) \leq \gamma \leq 10\,tol$.

```
Set δ = 0 and γ = ||A + A^T||_F/2.
WHILE γ > 10 max{tol, δ} DO
    Set α = sqrt(γ max{tol, δ})
    IF σ(H(α)) ∩ ıIR ≠ ∅ THEN
        set γ = α
    ELSE
        set δ = α
    END IF
END WHILE
END
```

Note that $\|A + A^T\|_F/2$ is a simple upper bound for $\beta(A)$. If $tol = 10^{-p}\|A + A^T\|_F/2$, then at most $\lceil \log_2 p \rceil$ bisection steps will be needed. We used Algorithm BISEC with $p = 12$ such that at most four bisection steps were required. The eigenvalues of $H(\alpha)$ were computed using DHABL(A)/DHASRD/DHAEVS(S) (symplectic scaling) and DGEEVX. A computed eigenvalue $\tilde{\lambda}$ was considered to be purely imaginary if

$$\mathrm{Re}\,(\tilde{\lambda}) < 10\varepsilon \|H(\alpha)\|_F |\tilde{\lambda}| =: tol_\lambda. \tag{13}$$

Let

$$A = U^T \mathrm{diag}\left(100, 99, \ldots, 3, \begin{bmatrix} \omega & 1 \\ -1 & \omega \end{bmatrix}\right) U$$

where $U = I_n - 2\frac{uu^T}{u^T u}$ and $u = [1, 2, \ldots, 100]^T$. Thus, $\beta(A) = \min\{3, \omega\}$. We computed the upper and lower bounds for $\beta(A)$, i.e., δ and γ, once by BISEC using DHABL(A)/DHASRD/ DHAEVS(S) and once by BISEC using DGEEVX. Table 10 shows the computed values for δ and γ as well as the required CPU times on a SUN Ultra-1-200E for several values of ω. The table shows that DHABL(A)/DHASRD/DHAEVS(S) is about twice as fast as DGEEVX. In view of (13) it is not surprising that both DHABL(A)/DHASRD/DHAEVS(S) and DGEEVX obtained the same bounds for $\beta(A)$.

	DHABL(A)/DHASRD/DHAEVS(S)			DGEEVX		
ω	δ	γ	CPU time	δ	γ	CPU time
10^{-1}	$1.84 \cdot 10^{-2}$	$1.03 \cdot 10^{-1}$	2.6	$1.84 \cdot 10^{-2}$	$1.03 \cdot 10^{-1}$	4.9
10^{-3}	$5.82 \cdot 10^{-4}$	$3.27 \cdot 10^{-3}$	2.5	$5.82 \cdot 10^{-4}$	$3.27 \cdot 10^{-3}$	5.0
10^{-5}	$3.27 \cdot 10^{-6}$	$1.84 \cdot 10^{-5}$	2.5	$3.27 \cdot 10^{-6}$	$1.84 \cdot 10^{-5}$	5.1
10^{-7}	$1.84 \cdot 10^{-8}$	$1.03 \cdot 10^{-7}$	2.5	$1.84 \cdot 10^{-8}$	$1.03 \cdot 10^{-7}$	5.1
10^{-9}	0.0	$3.27 \cdot 10^{-9}$	2.5	0.0	$3.27 \cdot 10^{-9}$	5.1

Table 10: Example 7, δ, γ, and CPU seconds of execution time.

Example 8 *$\mathcal{H}_\infty$–norm of a transfer matrix*
The computation of the $\mathcal{H}_\infty$–norm $\|G\|_{\mathcal{H}_\infty}$ of a *transfer matrix*

$$G(s) := C(sI - A)^{-1}B + D =: \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

where $A \in \mathbb{R}^{n\times n}$ is stable, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$, plays a central role in $\mathcal{H}_\infty$–control problems (see, e.g., [21, 20]). Here,

$$\|G\|_{\mathcal{H}_\infty} = \sup\{\, \|G(\imath s)\|_2 \;:\; s \in \mathbb{R}\,\}.$$

Let $\alpha \in \mathbb{R}$ such that $\alpha > \sigma_{\max}(D)$ where $\sigma_{\max}(D)$ denotes the largest singular value of D and define

$$\begin{aligned} F(\alpha) &= A + B(\alpha^2 I - D^T D)^{-1} D^T C, \\ R(\alpha) &= \frac{1}{\alpha^2} B(\alpha^2 I - D^T D)^{-1} B^T, \\ Q(\alpha) &= -C^T(I + D(\alpha^2 - D^T D)^{-1} D^T)C, \\ H(\alpha) &= \begin{bmatrix} F(\alpha) & R(\alpha) \\ Q(\alpha) & -F(\alpha)^T \end{bmatrix}. \end{aligned}$$

Obviously, $H(\alpha)$ is Hamiltonian.

It is shown in [20, 25] that if $\|D\| < \alpha$, then $\|G\|_{\mathcal{H}_\infty} < \alpha$ if and only if $\sigma(H(\alpha)) \cap \imath\mathbb{R} = \emptyset$. This result indicates that an upper and lower bound for $\|G\|_{\mathcal{H}_\infty}$ can be computed via Algorithm BISEC of Example 7. To get closer bounds, we also have to modify the stopping criterion in Algorithm BISEC. To obtain an interval with $\gamma \leq k\delta$, the stopping criterion is modified to $\gamma \leq k \max\{tol, \delta\}$. A similar bisection method was proposed in [9]. The initial search interval $[\delta, \gamma]$ may be chosen to be $[\,\sigma_{\max}(D),\ \gamma_{ub}\,]$ where γ_{ub} is the upper bound proposed in [9], i.e.,

$$\gamma_{ub} = \sigma_{\max}(D) + 2\sqrt{n \cdot \mathrm{trace}(W_c W_o)}$$

and W_c, W_o are the solutions of the controllability and observability Lyapunov equations

$$\begin{aligned} AW_c + W_c A^T + BB^T &= 0, \\ A^T W_o + W_o A + C^T C &= 0. \end{aligned}$$

As an example we computed upper and lower bounds for $\|G\|_{\mathcal{H}_\infty}$ for the state-space system given in Example 4 with $D = 0$. For this example, $\|G\|_{\mathcal{H}_\infty} \approx 2275.03$. The decision if $H(\alpha)$ has eigenvalues on the imaginary axis was based on the eigenvalues computed by DHAEVS analogous to Example 7. Using $k = 1.001$ (i.e., a relative accuracy of 0.1%), we obtained $2273.2 \leq \|G\|_{\mathcal{H}_\infty} \leq 2275.1$ after 15 iterations.

In [8], a quadratically convergent algorithm for the computation of the $\mathcal{H}_\infty$-norm is given. It requires the computation of all the purely imaginary eigenvalues of $H(\alpha)$ at each iteration step. These purely imaginary eigenvalues may conveniently be calculated DHABL/DHASRD/DHAEVS.

In addition to the above examples, DHABL/DHASRD/DHAEVS may be used to obtain the shifts for the multishift algorithm and other algorithms to solve algebraic Riccati equations [1, 2, 32, 34, 35].

6 Conclusions

We have presented Fortran 77 subroutines that transform Hamiltonian matrices to square-reduced form and approximate its eigenvalues as described in [14, 39]. Numerical experiments on computers with conventional architecture confirm the theoretical expectation that these subroutines do less than half the work and finish in less than half the time of the QR algorithm as implemented in LAPACK [4].

Rounding error analysis and numerical experiments also confirm the observation in [14, 39] that eigenvalues of larger magnitude are computed as accurately as their condition numbers suggest, but that tiny eigenvalues may be perturbed by a square root of the unit round.

Of the three scaling strategies tested here, norm scaling did not prove to be as effective as symplectic scaling and Hessenberg scaling. Although scaling sometimes improves accuracy, an error of order of the square root of machine precision can not always be avoided.

Acknowledgments

We thank the referees for helpful comments improving the readability of the manuscript and the functionality as well as modularity of the implemented subroutines.

References

[1] G. Ammar and P. Benner, *OSMARE: A Fortran 77 implementation of the orthogonal symplectic multishift method for continuous-time algebraic Riccati equations.* In preparation.

[2] G. Ammar, P. Benner, and V. Mehrmann, *A multishift algorithm for the numerical solution of algebraic Riccati equations*, Electr. Trans. Num. Anal., 1 (1993), pp. 33–48.

[3] G. Ammar and V. Mehrmann, *On Hamiltonian and symplectic Hessenberg forms*, Linear Algebra Appl., 149 (1991), pp. 55–72.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, PA, second ed., 1995.

[5] P. Benner, R. Byers, V. Mehrmann, and H. Xu, *Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices II.* In preparation.

[6] P. Benner, A. Laub, and V. Mehrmann, *A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case*, Tech. Rep. SPC 95_22, Fakultät für Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.

[7] 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 (1998), pp. 329–358.

[8] S. Boyd and V. Balakrishnan, *A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞-norm*, Sys. Control Lett., 15 (1990), pp. 1–7.

[9] S. Boyd, V. Balakrishnan, and P. Kabamba, *A bisection method for computing the $\mathcal{H}_\infty$ norm of a transfer matrix and related problems*, Math. Control, Signals, Sys., 2 (1989), pp. 207–219.

[10] N. Bruinsma and M. Steinbuch, *A fast algorithm to compute the H_∞-norm of a transfer function matrix*, Sys. Control Lett., 14 (1990), pp. 287–293.

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

[12] A. Bunse-Gerstner and V. Mehrmann, *A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation*, IEEE Trans. Automat. Control, AC-31 (1986), pp. 1104–1113.

[13] A. Bunse-Gerstner, V. Mehrmann, and D. Watkins, *An SR algorithm for Hamiltonian matrices based on Gaussian elimination*, Methods of Operations Research, 58 (1989), pp. 339–358.

[14] R. Byers, *Hamiltonian and Symplectic Algorithms for the Algebraic Riccati Equation*, PhD thesis, Cornell University, Dept. Comp. Sci., Ithaca, NY, 1983.

[15] ——, *A Hamiltonian QR-algorithm*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 212–229.

[16] ——, *A bisection method for measuring the distance of a stable to unstable matrices*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 875–881.

[17] E. Davison and W. Gesing, *The systematic design of control systems for the multivariable servomechanism problem*, in Alternatives for Linear Multivariable Control, M. Sain, J. Peczkowsky, and J. Melsa, eds., Nat. Eng. Consortium Inc., Chicago, IL, 1978.

[18] J. Dongarra, J. D. Croz, I. Duff, and S. Hammarling, *A set of Level 3 Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 16 (1990), pp. 1–17.

[19] J. Dongarra, J. D. Croz, S. Hammarling, and R. Hanson, *An extended set of FORTRAN Basic Linear Algebra Subprograms*, ACM Trans. Math. Soft., 14 (1988), pp. 1–17.

[20] J. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, *State-space solutions to standard H_2 and H_∞ control problems*, IEEE Trans. Automat. Control, 34 (1989), pp. 831–847.

[21] B. Francis, *A Course In $\mathcal{H}_\infty$ Control Theory*, vol. 88 of Lecture Notes in Control and Information Sciences, Springer–Verlag, Berlin, 1987.

[22] J. D. Gardiner, *A stabilized matrix sign function algorithm for solving algebraic Riccati equations*, SIAM J. Sci. Comput., 18 (1997), pp. 1393–1411.

[23] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, second ed., 1989.

[24] C. Kenney, A. Laub, and M. Wette, *A stability-enhancing scaling procedure for Schur-Riccati solvers*, Sys. Control Lett., 12 (1989), pp. 241–250.

[25] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995.

[26] A. Laub, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913–921.

[27] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, *Basic linear algebra subprograms for FORTRAN usage*, ACM Trans. Math. Software, 5 (1979), pp. 303–323.

[28] C. Paige and C. Van Loan, *A Schur decomposition for Hamiltonian matrices*, Linear Algebra Appl., 14 (1981), pp. 11–32.

[29] P. Pandey, *On scaling an algebraic Riccati equation*, in Proc. American Control Conf., San Francisco, CA, June 1993, pp. 1583–1587.

[30] B. Parlett and J. Le, *Forward instability of tridiagonal QR*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 279–316.

[31] B. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 296–304.

[32] R. Patel, Z. Lin, and P. Misra, *Computation of stable invariant subspaces of Hamiltonian matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 284–298.

[33] P. Petkov, N. Christov, and M. Konstantinov, *On the numerical properties of the Schur approach for solving the matrix Riccati equation*, Sys. Control Lett., 9 (1987), pp. 197–201.

[34] A. Raines and D. Watkins, *A class of Hamiltonian–symplectic methods for solving the algebraic Riccati equation*, Linear Algebra Appl., 205/206 (1994), pp. 1045–1060.

[35] V. Sima, *Algorithm GRICSR solving continuous-time algebraic Riccati equations using Gaussian symplectic transformations*, Stud. Res. Comp. Inf., 1 (1992), pp. 237–254.

[36] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines*—EISPACK *Guide*, vol. 6 of Lecture Notes in Computer Science, Springer-Verlag, New York, second ed., 1976.

[37] J. Sreedhar, P. V. Dooren, and A. Tits, *A fast algorithm to compute the real structured stability radius*, in Proc. Conf. Centennial Hurwitz on Stability Theory, Ticino (CH), May 21-26, 1995.

[38] ——, *A level-set idea to compute the real Hurwitz-stability radius*, in Proc. 34th IEEE Conf. Decision and Control, New Orleans, Louisiana, December 13-15, 1995, p. WA05.

[39] C. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 61 (1984), pp. 233–251.

[40] J. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

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

[42] H. Xu, *Solving Algebraic Riccati Equations via Skew-Hamiltonian Matrices*, PhD thesis, Inst. of Math., Fudan University, Shanghai, P.R. China, Apr. 1993.

[43] H. Xu and L. Lu, *Properties of a quadratic matrix equation and the solution of the continuous-time algebraic Riccati equation*, Linear Algebra Appl., 222 (1995), pp. 127–145.