Technische Universität Chemnitz-Zwickau

Sonderforschungsbereich 393

Numerische Simulation auf massiv parallelen Rechnern

Peter Benner Ralph Byers Eric Barth

HAMEV and SQRED: Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices Using Van Loan's Square Reduced Method

Preprint SFB393/96-06

Preprint-Reihe des Chemnitzer SFB 393

SFB393/96-06 **May 1996**

HAMEV and SQRED: Fortran 77 Subroutines for Computing the Eigenvalues of Hamiltonian Matrices Using Van Loan's Square Reduced Method

Peter Benner *
Fakultät für Mathematik
Technische Universität Chemnitz–Zwickau
09107 Chemnitz (FRG)
E-mail: benner@mathematik.tu-chemnitz.de

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

Eric Barth ‡
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012 (USA)
E-mail: barthe@cims.nyu.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 Hamiltonian uses only orthogonal symplectic similarity transformations. The eigenvalues can then be determined by applying the Hessenberg QR iteration to a matrix of half the order of the Hamiltonian matrix and taking the square roots of the computed values. Using scaling strategies similar to those suggested for algebraic Riccati equations can 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

*This author was supported by *Deutsche Forschungsgemeinschaft*, research grant *Me 790/7-1 Singuläre Steuerungsprobleme*.

†This author was partially supported by National Science Foundation grants INT-8922444, CCR-9404425, CCR-8820882, and DMS-9205538, and the Kansas Institute for Theoretical and Computational Science.

‡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.

Contents

1 Introduction **1**

2 Van Loan's Square Reduced Method **3**

3 Implementation **9**
3.1 Scaling 9
3.2 Eigenvalues on or close to the imaginary axis 10
3.3 Usage 11
3.4 Subroutine organization 12

4 Numerical Examples **14**

5 Applications **20**

6 Conclusions **23**

A The Fortran 77 Subroutine SQRED **24**
A.1 Subroutine description 24
A.2 Example program 27

B The Fortran 77 Subroutine HAMEV **31**
B.1 Subroutine description 31
B.2 Example program 35

C How to Obtain the Software **38**

References **39**

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. 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., [17, 8, 7] and the references given therein, and the problem of calculating the real and complex stability radius of a matrix [14, 36]. Hamiltonian matrices also have an intimate relationship 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 the symmetric solution matrix. Often, a *stabilizing* solution $\hat{X}$ of (2) is required in the sense that all the eigenvalues of $A - G\hat{X}$ 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 $\hat{X}$ of (2) is given by $\hat{X} = -WV^{-1}$. 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 [21].
Knowledge of approximations to the eigenvalues of the corresponding Hamiltonian matrix is crucial for some numerical solution methods for the CARE (2), e.g., for the multishift QR–like methods proposed in [2, 3, 33] and the algorithm presented in [31]. As suggested in [34], they can significantly improve the convergence of the SR algorithm [10] 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 [39, 40].

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 matrices with these properties must have the form (1). The eigenvalues of any matrix may be computed by the unsymmetric QR algorithm, see, e.g., [19]. But Hamiltonian matrices have a lot more structure than general unsymmetric matrices and it is desirable to exploit this structure in order to improve accuracy and to reduce computational cost as well as the required work space. A sound numerical procedure to compute the eigenvalues of any structured matrix might use only similarity transformations that are orthogonal and structure-preserving. For Hamiltonian matrices, structure is preserved by symplectic transformations. A matrix $S \in \mathbb{R}^{2n\times 2n}$ is called *symplectic* iff $S^T J S = J$ with J as in (3). 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 [13]. A structure preserving QR-like algorithm using SR decompositions has been presented in [10, 11, 34]. However, this algorithm suffers from numerical instabilities since the used symplectic transformations are not bounded in norm. Recently, an orthogonal symplectic method for computing the eigenvalues of a Hamiltonian matrix has been proposed [6]. 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 method proposed by Van Loan [37] uses the properties of the square of a Hamiltonian matrix. It is possible to reduce such a matrix to a Hessenberg-like form by structure-preserving similarity transformations. That is, the eigenvalues of the squared Hamiltonian matrix are computed by a strongly backward stable method. [1] 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 [37], it is shown that the eigenvalues computed by this method are exact eigenvalues of a Hamiltonian matrix $H + E$ where $\|E\| \leq O(\sqrt{\varepsilon})\|H\|$.

In the sequel, we will make use of 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 given by

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

and the *Frobenius* norm of A is defined by

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

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 will be denoted by I_n and the kth unit vector is given by e_k. Furthermore, we will make use of a notation also used in [19] for refering to a block of a matrix, that is, $A_{k:\ell,p:q}$ will denote the submatrix of A defined by its entries a_{ij}, $k \leq i \leq \ell$, $p \leq j \leq q$.

The outline of the paper is as follows. In the next section we review Van Loan's square reduced method and the necessary results for Hamiltonian matrices. 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. In Appendices A and B, we describe the input/output parameters of the Fortran 77 subroutines SQRED and HAMEV, give example programs, and in Appendix C, we show how to obtain the software.

[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 [9].

2 Van Loan's Square Reduced Method

At first we state some well-known results about Hamiltonian matrices which form the basis for Van Loan's method. These results can be found in [37]. We therefore omit the proofs. The spectrum of a Hamiltonian matrix has the following property.

Proposition 1 *Let $\lambda \in \sigma(H)$ where H 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 $-\lambda$, $\bar{\lambda}$, $-\bar{\lambda} \in \sigma(H)$.*

Therefore, the spectrum of any Hamiltonian matrix can be written as

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

where $\mathrm{Re}\,(\lambda_i) \geq 0$, $i = 1, \ldots, n$.
In many applications, the presence of purely imaginary eigenvalues of Hamiltonian matrices or the lack thereof plays an important role.

Theorem 2 *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 $\sigma(A - G\hat{X}) = \{ -\lambda_1, \ldots, -\lambda_n \}$.

In [27], the following condensed form of $2n \times 2n$ matrices is derived:

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*

$$\tilde{L} = U^T L U = \begin{bmatrix} \tilde{L}_{11} & \tilde{L}_{12} \\ \tilde{L}_{21} & \tilde{L}_{22} \end{bmatrix} = \begin{bmatrix} \diagdown & \square \\ \diagdown & \square \end{bmatrix}, \tag{5}$$

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

In the sequel, we will refer to the form (5) of a matrix as *Paige/Van Loan form* or *PVL form*. Now consider the square of a Hamiltonian matrix H as in (1),

$$K := H^2 = \begin{bmatrix} A^2 + GQ & AG - GA^T \\ QA - A^TQ & (A^2 + GQ)^T \end{bmatrix} =: \begin{bmatrix} K_1 & K_2 \\ K_3 & K_1^T \end{bmatrix}. \tag{6}$$

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, i.e., if $S \in \mathbb{R}^{2n\times 2n}$ is a symplectic matrix and H is Hamiltonian, then $S^{-1}H^2S$ is also a skew-Hamiltonian matrix. Therefore, using Theorem 3, the PVL form of a skew-Hamiltonian matrix $K = H^2$ is given by

$$\tilde{K} = U^T K U = \begin{bmatrix} \tilde{K_1} & \tilde{K_2} \\ 0 & \tilde{K_1}^T \end{bmatrix} = \begin{bmatrix} \diagdown & \square \\ & \diagdown \end{bmatrix}. \tag{7}$$

The eigenvalues of K can thus be computed by applying the QR iteration to $\tilde{K}_1$. Hence, according to (4), $\sigma(H)$ can be obtained by taking the positive and negative square roots of the computed eigenvalues of $\tilde{K}_1$.
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 (7).*

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, *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 [17, 8, 7] or the distance-to-instability problem [14, 36].
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 [2, 3, 31, 33] 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 [37]. 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 (7). 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 the following proposition [27].

Proposition 5 *If $U \in \mathbb{R}^{2n\times 2n}$ is orthogonal and symplectic, then*

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

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

It should further be noted that from Algorithm 4, the real Schur form of a skew-Hamiltonian matrix can be computed. 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. Observing that the block diagonal matrix

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

is again orthogonal and symplectic, we obtain the *skew-Hamiltonian Schur form* by

$$\hat{K} = V^T U^T K U V = \begin{bmatrix} T_1 & T_2 \\ 0 & T_1^T \end{bmatrix}. \tag{8}$$

The skew-Hamiltonian Schur form is used in the CARE solution methods proposed in [39, 40].

Before stating the implicit version of Algorithm 4, we have to take a closer look at orthogonal symplectic matrices. These can essentially be described by two classes of matrices, i.e., *symplectic Householder* and *symplectic Givens* matrices[2] [27]. These two matrix types can be defined as follows. Let $P = P(v) = I_n - 2\frac{vv^T}{v^T v}$, $v \in \mathbb{R}^n$, be a Householder matrix, then a *symplectic Householder matrix* is given by

$$P_s(v) = \begin{bmatrix} P(v) & 0 \\ 0 & P(v) \end{bmatrix}. \tag{9}$$

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

$$G_s(k, c, s) = \begin{bmatrix} C & -S \\ S & C \end{bmatrix}, \tag{10}$$

where $c, s \in \mathbb{R}$, $c^2 + s^2 = 1$, and $C = I_n + (c - 1)e_k e_k^T$, $S = s e_k e_k^T$.
Using the well-known abilities of Householder reflections and Givens rotations to annihilate a specific part of a vector (see, e.g., [19, 27]), and remembering Proposition 5, it is now possible to state the implicit version of Van Loan's algorithm.

[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.

Algorithm 6 (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 *transforming* H *to square-reduced form.* H *is overwritten by its square-reduced form* $U^T H U$.

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

   (a) *IF* $k \leq n-2$ *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	as dummy work space 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 SQRED 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., [19, 38].
Using flop[3] counts and estimates from [19], we get the following estimate for the computational cost of Algorithm 6.

	$4n^3$			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 $W = A^2 + GQ$,
	$7n^3$	+	$O(n^2)$	flops for the Hamiltonian QR iteration,
	n			flops for the square roots in Step 4.
total	$37n^3$	+	$O(n^2)$	flops

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 standard Hessenberg QR algorithm (as given, e.g., in [19]) applied to the Hamiltonian matrix H.

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 6. Then Van Loan proves in [37] that

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

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

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

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

Remark 7 *The perturbation E can be considered as a structured perturbation, that is, if the algorithm is implemented carefully, E is a Hamiltonian matrix.*

Using some heuristic, the following result for a simple eigenvalue computed by Algorithm SQRED (implicit or explicit version) is derived in [37].

$$|\lambda-\tilde{\lambda}| \approx \min\left\{\frac{\varepsilon\|H\|_2^2}{s(\lambda)|\lambda|}, \frac{\sqrt{\varepsilon}\|H\|_2}{s(\lambda)}\right\} = \varepsilon\frac{\|H\|_2}{s(\lambda)} \times \min\left\{\frac{\|H\|_2}{|\lambda|}, \frac{1}{\sqrt{\varepsilon}}\right\} \tag{11}$$

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.

[3]Following [19], we define each floating point arithmetic operation together with the associated integer indexing as a flop.

The estimate (11) can be interpreted using the standard estimate for eigenvalues λ^{QR} computed by the Hessenberg QR iteration [38],

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

as follows:

- If $\|H\|_2/\lambda \approx 1$, then $\tilde{\lambda}$ is as accurate as λ^{QR},
- If $\|H\|_2/|\lambda| > 1$, 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 the matter. In the mentioned multishift and related algorithms [2, 3, 33, 31], more accurate eigenvalues can be obtained during the iteration. Sometimes, accurate shifts may even lead to forward instability, see [29]. For a discussion of the problem of eigenvalues close to the imaginary axis (including "small" eigenvalues) see Section 3.2.
As it was observed in [14], it is safe to use Van Loan's square reduced method when computing the stability radius of a real matrix. The same argument can also be applied for the application of Algorithm 6 to the $\mathcal{H}_\infty$–norm computation, see Section 5.

3 Implementation

For the actual implementation we chose the implicit version of the algorithm in order to avoid the need of explicitly computing the square of the Hamiltonian matrix. The subroutine SQRED implements the first step of Algorithm 6, i.e., computes the square-reduced Hamiltonian and, if required, the orthogonal symplectic transformation matrix. The subroutine HAMEV performs steps 2.-4. of Algorithm 6 using subroutine SQRED. We chose to separate these steps in order to make it possible to use the square-reduced Hamiltonian matrix itself. The implementation and documentation of all subroutines follow the standards proposed for the **S**ubroutine **LI**brary in **C**ontrol and **S**ystems **T**heory SLICOT (see [25]). Besides the subroutines described below we used the DOUBLE PRECISION versions of the BLAS (levels 1 and 2) [23, 16] and LAPACK [4].

We will now describe some implementation details and how the subroutines are used.

3.1 Scaling

It is possible to scale or balance a matrix by a diagonal similarity transformation to make all row norms and all columns (approximately) equal. Balancing tends to reduce the ill-effects of rounding error in subsequent eigenvalue calculations [30, 26]. However, the balancing procedure described in [30] destroys Hamiltonian structure. We need a strategy that uses symplectic similarity transformations only. Diagonal symplectic matrices take the form

$$D_s = \begin{bmatrix} D & 0 \\ 0 & D^{-1} \end{bmatrix} \tag{12}$$

where D is diagonal. In this section we describe some balancing strategies.

A simple strategy is to balance A using a diagonal matrix $D = D_A$ computed by a packaged, unstructured balancing subroutine like DGEBAL from LAPACK [4]. The diagonal similarity transformations D_A can then be extended to a symplectic diagonal similarity D_s as in (12). Of course, the choice of D_s ignores G and Q, so it is unlikely to be optimal.

Another possibility is to use a scaling strategy known to improve the accuracy of solution methods for the CARE (2). Here, the diagonal matrix D in (12) is chosen as $D_\rho = I_n/\sqrt{\rho}$ for some scalar $\rho \neq 0$. This strategy results in a Hamiltonian matrix

$$\tilde{H} := \begin{bmatrix} A & \tilde{G} \\ \tilde{Q} & -A^T \end{bmatrix} := \begin{bmatrix} {D_\rho}^{-1} & 0 \\ 0 & D_\rho \end{bmatrix} \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \begin{bmatrix} D_\rho & 0 \\ 0 & {D_\rho}^{-1} \end{bmatrix} \tag{13}$$

where $\tilde{G} = \rho G$ and $\tilde{Q} = Q/\rho$. It was observed in [32] 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). The choice $\rho = \sqrt{\|Q\|/\|G\|}$ gives $\|\tilde{G}\| = \|\tilde{Q}\|$, but, in general, it is not possible to choose ρ to achieve the "optimal" scaling $\|A\| = \|\tilde{G}\| = \|\tilde{Q}\|$.

Both ideas can be combined. 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

$$\tilde{H} = \begin{bmatrix} D_A^{-1} A D_A & \rho D_A^{-1} G D_A^{-1} \\ \frac{1}{\rho} D_A Q D_A & -D_A A^T D_A^{-1} \end{bmatrix} \tag{14}$$

where $\rho = \sqrt{\|D_A Q D_A\|_1 / \|D_A^{-1} G D_A^{-1}\|_1}$.
Closely related to the above scaling via $D = \rho I$ is the strategy proposed in [20, 28] 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 (12) where $D = \sqrt{\tau} I_n$. We will refer to this scaling strategy as *norm scaling*.
In order not to introduce rounding errors by scaling, we choose in both scaling strategies the parameters ρ and τ among the set of numbers β^σ where β denotes the base of the floating point number system and σ is a signed integer. Note that the LAPACK subroutine DGEBAL is not implemented in this fashion such that rounding errors may well be introduced when scaling A in our symplectic scaling strategy.
The special case in which Q is of small magnitude arises naturally during defect correction of solutions of the CARE [24]. In this case, Q tends to suffer from subtractive cancelation and is known only to low relative error. The diagonal balancing strategy tends to exaggerate the ill-effect of the errors in a small magnitude, low accuracy Q. To avoid this the balancing strategies bound ρ and τ from below by 1 and from above to prevent overflow in the computation of ρG and τ.
In Step 3. of Algorithm 6, the usual balancing as proposed in [30] of a matrix can be applied to W. This can be achieved, as mentioned above, using the 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.

When calling HAMEV, the following choices for the input parameter `JOBSCL` are possible:

A – symplectic scaling and Hessenberg scaling,
B – norm scaling and Hessenberg scaling,
N – no scaling.

For any other choice, only Hessenberg scaling is performed.

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 [8, 7, 17] or the stability radius of real and complex matrices [14, 36] have to decide whether there are eigenvalues on the imaginary axis or not. The knowledge of the spectrum of H also gives information about the solution of the CARE (2). 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 [21]. 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, 33, 31].

The subroutine HAMEV can be used to decide if a Hamiltonian matrix has eigenvalues on the imaginary axis in the following way. If the input argument `ORDER` is set to 'O' or 'o', then all eigenvalues satisfying

$$|\mathrm{Re}(\lambda)| \leq tol \times |\lambda| \tag{15}$$

(where *tol* is a user-defined tolerance) are placed at the end of $\{\lambda_1, \ldots, \lambda_n\}$. That is, if *npi* is the number of computed eigenvalues satisfying condition (15), then this number is returned by HAMEV and those eigenvalues within the given relative tolerance to the imaginary axis are returned as $\lambda_{n-npi+1}, \ldots, \lambda_n$ (and $-\lambda_{n-npi+1}, \ldots, -\lambda_n$, respectively). The final decision which of these eigenvalues is considered to be purely imaginary is still left to the user. This decision can, for instance, be made by the choice of *tol*. 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.

As default tolerance we choose $tol = \sqrt{\varepsilon}$ where ε is the machine precision. This is only used if on input to HAMEV, we have for the corresponding argument `TOL` < 0. This tolerance is inspired by the error estimate given in (11). A worst case analysis shows that the real part of the computed analogue of a purely imaginary eigenvalue can have a real part of order $O(\sqrt{\varepsilon})$.

3.3 Usage

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

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

SQRED can be used by itself to reduce a Hamiltonian matrix H to square-reduced form and is also used by the subroutine HAMEV. The subroutine is called as follows:

```
CALL SQRED(N, A, LDA, GP, QP, U, LDU, RWORK, COMPU, IERR)
```

For a complete description of input and output arguments see Appendix A. The Hamiltonian matrix is provided by its blocks A, G, and Q, where the symmetric matrices G and Q are stored in the arrays `GP`, `QP` using lower packed storage mode (see [4]). That is, only the lower triangles of G and Q are stored columnwise.

If required (as defined by `COMPU`), the orthogonal symplectic transformations are accumulated in a matrix U. The argument `U` contains the first n rows of this matrix which completely define the orthogonal symplectic matrix U due to Proposition 5.

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

The subroutine HAMEV computes the eigenvalues of the Hamiltonian matrix. It is possible to obtain all $2n$ eigenvalues or the n eigenvalues with nonnegative real parts (unstable eigenvalues) or nonpositive real parts (stable eigenvalues).
The calling sequence is

```
 CALL HAMEV(N, A, LDA, GP, QP, U, LDU, SCALE, NPI, WR, WI, RWORK,
$           TOL, COMPU, JOBEV, JOBSCL, ORDER, IERR)
```

For a complete description of input and output arguments see Appendix B. After a possible symplectic or norm scaling as proposed in Section 3.1, the Hamiltonian matrix and the orthogonal symplectic transformations are treated exactly as in SQRED. That is, after returning from HAMEV, the Hamiltonian matrix is the square-reduced Hamiltonian computed by SQRED. The eigenvalues are returned in `WR` (real parts) and `WI` (imaginary parts). The arguments `NPI`, `TOL`, and `ORDER` refer to the ordering of the eigenvalues as explained in Section 3.2 whereas `JOBSCL` defines which of the scaling strategies of Section 3.1 is used. The argument `SCALE` provides all necessary scaling parameters to retrieve the Hamiltonian input matrix from the output matrix. **Note** that the eigenvalues of the Hamiltonian input matrix are returned. If norm scaling is performed, these are not the eigenvalues of the Hamiltonian output matrix. In that case, the eigenvalues of the Hamiltonian output matrix $\hat{H}$ are $\hat{\lambda}_i = (\texttt{WR(I)} + \imath\texttt{WI(I)})/\texttt{SCALE(1)}$ and the Hamiltonian matrix corresponding to the returned eigenvalues can be retrieved by setting $H = \texttt{SCALE(1)} \times \hat{H}$.

3.4 Subroutine organization

The basis for the subroutines SQRED and HAMEV are the BLAS and LAPACK [23, 16, 4]. Table 1 shows the necessary functions and subroutines from these subroutine libraries.

BLAS routines		LAPACK routines			
Level 1	Level 2	auxiliary			computational
DAXPY	DGEMV	DLABAD	DLANSP	DLASET	DGEBAL
DCOPY	DSPMV	DLACPY	DLANV2	DLASSQ	DHSEQR
DDOT	DGER	DLAHQR	DLAPY2	DRSCL	
DNRM2	DSPR2	DLAE2	DLARFX	ILAENV	
DROT		DLAMCH	DLARFG	LSAME	
DSCAL		DLANGE	DLARTG	XERBLA	
DSWAP		DLANHS	DLASCL		
IDAMAX					

Table 1: Necessary BLAS and LAPACK routines

Besides the BLAS and LAPACK routines, we need the following subroutines:

CROOT : Computes 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 [35] subroutine CSROOT.

SYREF : Performs a similarity transformation of a symmetric matrix given in lower packed storage mode with a Householder matrix as proposed in [38].

HAMGIV : This subroutine performs a similarity transformation of a Hamiltonian matrix supplied by its blocks A, G, and Q as described in Section 3.3 with a symplectic Givens rotation matrix $G_s(k, c, s)$ as in (10).

Givens rotations were generated by DLARTG and applied to H by HAMGIV whereas accumulating the orthogonal symplectic transformations uses DROT. Householder reflections were generated by DLARFG and applied to A by DLARFX and to G, Q by SYREF. Accumulating the reflections was done by DLARFX. Figure 1 shows the subroutine hierarchy.

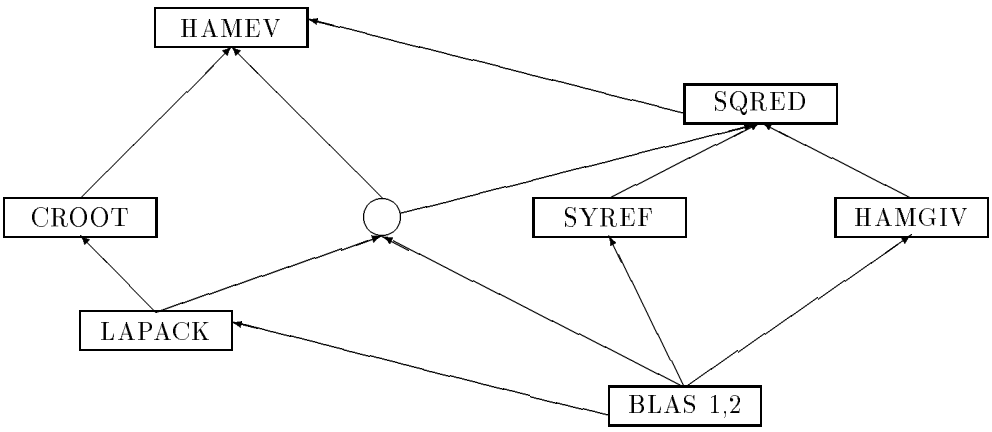

Figure 1: Subroutine hierarchy

4 Numerical Examples

We tested our subroutines for all Hamiltonian matrices from the benchmark collection for continuous-time algebraic Riccati equations [5], the examples given in [37], and some randomly generated examples. Here, we present the most interesting results obtained by these experiments.

The numerical tests were performed using IEEE double precision arithmetic with machine precision $\varepsilon \approx 2.2204 \times 10^{-16}$ on Hewlett Packard apollo series 700 workstations with operating system HP-UX 9.0x (for several values of x) and on a SUN SPARCstation 10 with operating system SunOS 4.1. The compilers were the HP-UX and SUN Fortran 77 compilers invoked by `f77` and only low level optimization was allowed.

We compared our subroutines with the LAPACK driver routine DGEEVX for computing the eigenvalues of an unsymmetric matrix using the QR algorithm. In all given tables, HAMEV(X) denotes calling HAMEV with the scaling strategy X as given in Section 3.1. (Here, 'H' will denote Hessenberg scaling.)

Example 1 [37, 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, $\sigma(H) = \{\pm 1, \pm 10^{-2}, \pm 10^{-4}, \pm 10^{-6}, \pm 10^{-8}\}$. Table 2 shows the absolute errors in the eigenvalue approximations computed by HAMEV and DGEEVX. In this example, choosing the scaling parameters from the set of integer powers of the machine base results in scaling parameters all equal to one. Thus, the results for all scaling strategies in this example were the same.

λ	HAMEV	DGEEVX
1	$1.2 \cdot 10^{-15}$	$1.4 \cdot 10^{-15}$
10^{-2}	$1.0 \cdot 10^{-17}$	$2.9 \cdot 10^{-17}$
10^{-4}	$1.3 \cdot 10^{-14}$	$1.8 \cdot 10^{-18}$
10^{-6}	$1.7 \cdot 10^{-14}$	$1.7 \cdot 10^{-18}$
10^{-8}	$4.3 \cdot 10^{-11}$	$8.0 \cdot 10^{-18}$

Table 2: Example 1, absolute errors $|\lambda - \tilde{\lambda}|$

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 HAMEV varied by about one order of magnitude for the smaller eigenvalues. The values given above can be considered as the geometrical mean of the absolute errors for these test runs.

Example 2 [37, 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 eigenvalue condition number becomes worse for the smaller eigenvalues. A Hamiltonian matrix having the same eigenvalues as the Frank matrix together with their negative counterparts is generated as in Example 1,

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

with $U \in \mathbb{R}^{2n\times 2n}$ orthogonal symplectic randomly generated by n symplectic rotations and n reflectors.
We tested our subroutines for $n = 12$. Since exact eigenvalues are not known, we compare the values computed by HAMEV with those obtained by DGEEVX (denoted by λ^{QR}). The results are given in Table 3 where only the results for the 5 eigenvalues of smallest absolute value (and worst condition number) are shown.

$\lambda \approx$	$s(\lambda)$	HAMEV(A)	HAMEV(B)	HAMEV(H)	HAMEV(N)
0.2847	$1.8 \cdot 10^{-6}$	$9.1 \cdot 10^{-10}$	$3.5 \cdot 10^{-8}$	$9.1 \cdot 10^{-10}$	$1.7 \cdot 10^{-10}$
0.1436	$1.8 \cdot 10^{-6}$	$5.8 \cdot 10^{-9}$	$1.0 \cdot 10^{-6}$	$5.8 \cdot 10^{-9}$	$6.7 \cdot 10^{-8}$
0.0812	$3.8 \cdot 10^{-8}$	$1.2 \cdot 10^{-7}$	$7.0 \cdot 10^{-6}$	$1.2 \cdot 10^{-7}$	$5.4 \cdot 10^{-7}$
0.0495	$2.6 \cdot 10^{-8}$	$3.9 \cdot 10^{-7}$	$1.4 \cdot 10^{-5}$	$3.9 \cdot 10^{-7}$	$1.4 \cdot 10^{-6}$
0.0310	$5.5 \cdot 10^{-8}$	$3.4 \cdot 10^{-7}$	$8.7 \cdot 10^{-6}$	$3.4 \cdot 10^{-7}$	$1.0 \cdot 10^{-6}$

Table 3: Example 2, $|\lambda^{HAMEV} - \lambda^{QR}|$

Here, all scaling parameters for symplectic scaling are equal to one and thus, the results for HAMEV(A) are equal to those for HAMEV(H). Denoting the minimum singular values of $A - \tilde{\lambda}I$ for the computed eigenvalues $\tilde{\lambda}$ by $\tilde{\sigma}_{\min}$, we obtain $\tilde{\sigma}_{\min} \approx 10\varepsilon$ for DGEEVX, whereas for HAMEV(X) (X = A,H,N), $\tilde{\sigma}_{\min}$ is at most one order of magnitude larger whereas for HAMEV(B), $\tilde{\sigma}_{\min}$ is up to two orders of magnitude larger than for DGEEVX. This shows that for very sensitive eigenvalues, norm scaling can decrease the accuracy of the computed eigenvalues.

Example 3 [5, Eample 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 $\{\pm j\}$ with algebraic multiplicity two and geometric multiplicity one. The computed eigenvalues are

- for HAMEV with symplectic scaling,
  $\{ +.8407319236349208E-08 \pm .1000000000000000E+01i$
  $\quad -.8407319236349208E-08 \pm .1000000000000000E+01i \}$,
- for HAMEV with norm scaling,
  $\{ 0 \pm .9999999480808375E+00i,$
  $\quad 0 \pm .1000000051919159E+01i \}$,
- for HAMEV with Hessenberg scaling,
  $\{ 0 \pm .9999999913700510E+00i,$
  $\quad 0 \pm .1000000008629949E+01i \}$,
- for HAMEV without any scaling,
  $\{ 0 \pm .9999999913700510E+00i,$
  $\quad 0 \pm .1000000008629949E+01i \}$,
- for DGEEVX,
  $\{ +.3682987155528838E-07 \pm .1000000016020978E+01i,$
  $\quad -.3682986875631206E-07 \pm .9999999839790201E+00i \}$.

The different scaling strategies for HAMEV result in different perturbations of the eigenvalues. For symplectic scaling, the imaginary part is computed correctly. For the other scaling strategies, the computed eigenvalues have real part exactly zero whereas the imaginary part is perturbed. The QR algorithm as implemented in DGEEVX computes eigenvalues that are perturbed along the real axis as well as along the imaginary axis. In particular, they do not appear in plus-minus pairs as the exact eigenvalues do. Also, the errors are one order of magnitude larger than for the eigenvalues computed by HAMEV (except for norm scaling). We can conclude that in this example, symplectic scaling yields the best result since it returns the right pairing and algebraic multiplicity of the eigenvalues and the eigenvalues closest to the correct ones.

Example 4 [5, Example 6], [15] This example comes from a control problem for a J—100 jet engine as special case of a multivariable servomechanism problem. The system is described by

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

where the state vector x contains the state of the jet engine, the actuators, and the sensors. For the system matrices $A \in \mathbb{R}^{30\times 30}$, $B \in \mathbb{R}^{30\times 3}$, $C \in \mathbb{R}^{5\times 30}$ we refer to [5, 15]. The corresponding Hamiltonian matrix is

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

We know the exact values only for four of the eigenvalues of H which are 33.3 and a triple eigenvalue at 20.0. We give absolute errors for these eigenvalues as computed by HAMEV

(Table 4) as well as the difference between the values computed by HAMEV (denoted by λ^{HAMEV}) and DGEEVX (denoted by λ^{QR}) for the eigenvalues of smallest and largest modulus (Table 5).

$\lambda =$	HAMEV(A)	HAMEV(B)	HAMEV(H)	HAMEV(N)	DGEEVX
33.3	$1.1 \cdot 10^{-9}$	$5.4 \cdot 10^{-10}$	$3.4 \cdot 10^{-11}$	$3.5 \cdot 10^{-11}$	0
20.0	$1.4 \cdot 10^{-9}$	$1.2 \cdot 10^{-11}$	$6.2 \cdot 10^{-10}$	$5.9 \cdot 10^{-10}$	0
20.0	$1.4 \cdot 10^{-9}$	$1.2 \cdot 10^{-11}$	$6.2 \cdot 10^{-10}$	$5.9 \cdot 10^{-10}$	0
20.0	$1.9 \cdot 10^{-10}$	$6.3 \cdot 10^{-12}$	$6.5 \cdot 10^{-11}$	$1.6 \cdot 10^{-12}$	0

Table 4: Example 4, absolute errors for exact eigenvalues

$\lambda \approx$	HAMEV(A)	HAMEV(B)	HAMEV(H)	HAMEV(N)
577.036	$1.6 \cdot 10^{-12}$	$4.3 \cdot 10^{-11}$	$4.0 \cdot 10^{-11}$	$4.1 \cdot 10^{-11}$
0.182	$8.6 \cdot 10^{-13}$	$3.2 \cdot 10^{-15}$	$1.3 \cdot 10^{-10}$	$1.3 \cdot 10^{-10}$

Table 5: Example 4, $|\lambda^{HAMEV} - \lambda^{QR}|$

Tables 4, 5 show that norm scaling can in some cases improve the accuracy of eigenvalues computed by the square reduced method significantly. Symplectic scaling also improves the accuracy for the eigenvalues of smallest and largest modulus.

Example 5 [22, Example 5], [5, Example 15], [37, Example 1] The Hamiltonian matrix in this problem comes from the position and velocity control problem for a string of N high-speed 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

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

The Hamiltonian matrix is

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

with $n = 2N - 1$. For all $\lambda \in \sigma(H)$ and for all tested orders $N = 5k$, $k = 1, \ldots, 20$, the eigenvalues computed by HAMEV for all scaling strategies satisfied

$$|\lambda^{QR} - \lambda^{HAMEV}| \approx \varepsilon \|H\|$$

where λ^{QR} denotes the eigenvalues computed by LAPACK subroutine DGEEVX. All eigenvalues are well conditioned and $\|H\|_2/|\lambda| = O(1)$ for all $\lambda \in \sigma(H)$.
In Table 6 we give the CPU times used by HAMEV and DGEEVX on a SUN SPARCstation 10. The timings for HAMEV are similar for all scaling strategies and we therefore give only the values obtained by performing only Hessenberg scaling.

N	25	50	75	100
HAMEV	0.28	1.80	5.09	11.42
DGEEVX	1.34	8.53	29.47	60.24

Table 6: Example 5, CPU times (seconds)

From Table 6 we see that HAMEV is significantly faster than DGEEVX. For increasing N, the CPU times used by HAMEV tend to be less than 20% of that of DGEEVX. This is even faster than predicted by the flop count. In all the tested computing environments, matrix-vector products where some blocks in the matrix or vector are zero perform much faster than for arbitrary matrices/vectors. Since in this example, $GQ = 0$, $QA - A^TQ$ is tridiagonal, and $A^2 = A^2 + GQ$ is already in upper Hessenberg form, many of the operations in Algorithm 6 are products involving zero blocks. Therefore, the unexpected speed-up in HAMEV can be explained by this effect due to the efficient implementation of multiplication with zeros on the tested computers.

Example 6 We tested our subroutines for randomly generated Hamiltonian matrices with entries distributed normally in the interval $[-1, 1]$. Since the eigenvalue distribution for these examples usually behaves nicely, the eigenvalues computed by HAMEV are as accurate as for DGEEVX. We give the CPU times for $2n \times 2n$ examples for several sizes of n. For each size of n, we computed 100 examples. The values given in Table 7 are the mean values of the CPU times measured on a SUN SPARCstation 10.

n	25	50	75	100	125	150	175	200
HAMEV	0.10	0.65	2.08	4.63	8.77	14.78	22.85	33.49
DGEEVX	0.26	1.59	4.97	11.28	21.83	37.73	59.25	87.43

Table 7: . Example 6, CPU times (seconds)

Table 7 shows that the speed up of HAMEV compared with DGEEVX is a little smaller than expected from the flop counts. This is due to the more complicated data structure,

memory access, and index handling of SQRED. For growing n, the computational time used by HAMEV tends to be about 38% of that required by DGEEVX (as opposed to the expected 36% from the flop count).
Besides the faster computation of the eigenvalues HAMEV returns the right pairing of the eigenvalues as $\pm\lambda_i, i = 1, \ldots, n$. Since DGEEVX treats a Hamiltonian matrix like an arbitrary unsymmetric matrix, small perturbations can cause computed eigenvalues with small real parts to cross the imaginary axis. For example, the number of stable eigenvalues returned by DGEEVX for $n = 100$ varied between 95 and 104.

5 Applications

To show some possible applications for our subroutines we chose two problems from control theory. These are considered in Examples 7 and 8. Another possible application is to use HAMEV to approximate the eigenvalues of a Hamiltonian matrix and to use them as shifts in a solver for CAREs (2) such as the methods proposed in [2, 3, 31, 33, 34]. In [1] it will be reported how our subroutines are used during the numerical solution of the CARE (2) by the multishift algorithm as prosposed in [2].

Example 7 *The real stability radius of a real matrix* [14]
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. In other words, *how large must a perturbation be to make A 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 [14], a bisection method for measuring $\beta(A)$ is presented which is based on the following observation:
Let $\alpha \geq 0$. 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: $\delta, \gamma \in \mathbb{R}$ such that 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)) ∩ ıR ≠ ∅ 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 required. We used Algorithm BISEC with $p = 12$ such that at most four bisection steps were required. The eigenvalues of $H(\alpha)$ were computed by HAMEV and DGEEVX and the decision if there exist purely imaginary eigenvalues was based on the relative tolerance

$$\mathrm{Re}(\lambda) < 10\varepsilon\|H(\alpha)\|_F|\lambda| =: tol_\lambda. \tag{16}$$

Let

$$A = U^T \begin{bmatrix} 100 & & & & & \\ & 99 & & & & \\ & & \ddots & & & \\ & & & 3 & & \\ & & & & \omega & 1 \\ & & & & -1 & \omega \end{bmatrix} U$$

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 HAMEV and once by BISEC using DGEEVX. The number of computed purely imaginary eigenvalues with respect to tol_λ is returned by HAMEV if the subroutine is called with ORDER = 'O' and TOL $= 10\varepsilon\|H(\alpha)\|_F$. In the version using DGEEVX, the criterion (16) is checked for the returned eigenvalues. Table 8 shows the computed values for δ and γ as well as the required CPU times on a SUN SPARCstation 10 for several values of ω.

	HAMEV			DGEEVX		
ω	δ	γ	CPU time	δ	γ	CPU time
10^{-1}	$1.84 \cdot 10^{-2}$	$1.03 \cdot 10^{-1}$	20.30	$1.84 \cdot 10^{-2}$	$1.03 \cdot 10^{-1}$	39.73
10^{-3}	$5.82 \cdot 10^{-4}$	$3.27 \cdot 10^{-3}$	19.57	$5.82 \cdot 10^{-4}$	$3.27 \cdot 10^{-3}$	40.52
10^{-5}	$3.27 \cdot 10^{-6}$	$1.84 \cdot 10^{-5}$	19.92	$3.27 \cdot 10^{-6}$	$1.84 \cdot 10^{-5}$	40.23
10^{-7}	$1.84 \cdot 10^{-8}$	$1.03 \cdot 10^{-7}$	19.62	$1.84 \cdot 10^{-8}$	$1.03 \cdot 10^{-7}$	41.61
10^{-9}	0.0	$3.27 \cdot 10^{-9}$	19.41	0.0	$3.27 \cdot 10^{-9}$	41.54

Table 8: Example 7, δ, γ, and CPU times (seconds)

The computed bounds for $\beta(A)$ are independent of the chosen method for the decision if $H(\alpha)$ has eigenvalues on the imaginary axis. Note that in [14] it was shown that it is safe to base the decision if $H(\alpha)$ has eigenvalues on the imaginary axis on any method preserving the Hamiltonian structure of $H(\alpha)$ which Van Loan's method does as observed in Remark 7. Here, the method using HAMEV is about twice as fast as the method using DGEEVX. This is a little less than for the randomly generated examples.

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., [18, 17]). 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

$$F(\alpha) \;=\; A + B(\alpha^2 I - D^T D)^{-1} D^T C,$$

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

Obviously, $H(\alpha)$ is Hamiltonian. The following result (e.g., [41]) can be used to approximate $\|G\|_{\mathcal{H}_\infty}$.

$$\|G\|_{\mathcal{H}_\infty} < \alpha \iff \sigma_{\max}(D) < \alpha \text{ and } \sigma(H(\alpha)) \cap \imath\mathbb{R} = \emptyset$$

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 [8]. The initial search interval $[\delta,\ \gamma]$ can naively be chose to be $[\,\sigma_{\max}(D),\ \gamma_{ub}\,]$ where γ_{ub} is the upper bound proposed in [8], i.e.,

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

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 HAMEV 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 (1.24 seconds on a HP 712).
In [7], a quadratically convergent algorithm for the computation of the $\mathcal{H}_\infty$-norm is given that requires the computation of all the purely imaginary eigenvalues of $H(\alpha)$ in each iteration step. Thus, our subroutines can also be used in this algorithm.

6 Conclusions

We have presented Fortran 77 subroutines that transform Hamiltonian matrices to square-reduced form and approximate its eigenvalues as described in [12, 37]. 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 [12, 37] 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. Sometimes scaling improves accuracy, but an error of order of the square root of machine precision can not always be avoided.

A The Fortran 77 Subroutine SQRED

This section describes the subroutine SQRED and gives an example program showing how to use SQRED. The description and example program text follow the SLICOT subroutine documentation standard [25].

A.1 Subroutine description

1. Purpose

To transform a Hamiltonian matrix

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \tag{17}$$

into a square-reduced Hamiltonian matrix

$$\hat{H} = \begin{bmatrix} \hat{A} & \hat{G} \\ \hat{Q} & -\hat{A}^T \end{bmatrix} \tag{18}$$

by an orthogonal symplectic similarity transformation $\hat{H} = U^T H U$ where

$$U = \begin{bmatrix} U_1 & U_2 \\ -U_2 & U_1 \end{bmatrix}. \tag{19}$$

2. Specification

```
      SUBROUTINE       SQRED(N, A, LDA, GP, QP, U, LDU, RWORK, COMPU,
     1                       IERR)
      INTEGER          N, LDA, LDU, IERR
      DOUBLE PRECISION A(LDA,N), GP(N*(N+1)/2), QP(N*(N+1)/2), U(LDU,2*N),
     1                 RWORK(2*N)
      CHARACTER        COMPU
```

3. Argument List

3.1. Arguments In

N – INTEGER.

The order of the matrices A, G, and Q.
N $\geq$ 1.

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N×N part of this array contains the upper left block A of the Hamiltonian matrix H in (17).
Note that this array is overwritten.

LDA – INTEGER.

The leading dimension of array A as declared in the calling program.
LDA $\geq$ N.

GP, QP – DOUBLE PRECISION arrays of DIMENSION at least N*(N+1)/2.

Array GP contains the upper right symmetric block G and array QP the lower left symmetric block Q of the Hamiltonian matrix H in (17) in lower packed storage mode, i.e., the lower triangles of the symmetric matrices are stored by columns.

Note that these arrays are overwritten.

U – DOUBLE PRECISION array of DIMENSION (LDU,*udim*).

If COMPU = 'A' or 'a' or 'F' or 'f', then $udim \geq 2*N$. Otherwise, $udim \geq 1$.
If COMPU = 'A' or 'a', the leading N×2N part of this array **must** contain the first N rows of an orthogonal symplectic matrix. Otherwise U is not referenced on input.

Note that this array is overwritten if COMPU = 'A' or 'a' or 'F' or 'f'.

LDU – INTEGER.

The leading dimension of array U as declared in the calling program.
If COMPU = 'A' or 'a' or 'F' or 'f', then LDU ≥ N. Otherwise, LDU ≥ 1.

3.2. Arguments Out

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N×N part of this array contains the upper right block $\hat{A}$ of the square-reduced Hamiltonian matrix $\hat{H}$ in (18).

GP, QP – DOUBLE PRECISION arrays of DIMENSION at least N*(N+1)/2.

Array GP contains the upper right symmetric block $\hat{G}$ and array QP the lower left symmetric block $\hat{Q}$ of the square-reduced Hamiltonian matrix $\hat{H}$ in (18) in lower packed storage mode, i.e., the lower triangles of the symmetric matrices are stored by columns.

U – DOUBLE PRECISION array of DIMENSION (LDU,*udim*).

If COMPU = 'A' or 'a' or 'F' or 'f', then $udim \geq 2*N$. Otherwise, $udim \geq 1$.
If COMPU = 'F' or 'f', then array U contains the first N rows of the orthogonal symplectic matrix U in (19).
IF COMPU = 'A' or 'a', then array U contains the first N rows of the product $\tilde{U}U$ of an orthogonal symplectic input matrix $\tilde{U}$ and U from (19).
Otherwise, U is not referenced.

3.3. Work space

RWORK – DOUBLE PRECISION array of DIMENSION at least 2*N.

3.4. Tolerances

None.

3.5. Mode Parameters

COMPU – CHARACTER.

Indicates whether the orthogonal symplectic matrix U of (19) is returned or accumulated into an orthogonal symplectic matrix or if the transformation matrix is not required.

COMPU = 'A' or 'a', (The orthogonal symplectic similarity transformations are accumulated in U, i.e., U is the product of an orthogonal symplectic input matrix and the similarity transformation matrix of (19). On input, array U **must** contain the first N rows of an orthogonal symplectic 2N×2N matrix);

COMPU = 'F' or 'f', (The matrix U in (19) is formed, i.e., the first N rows of the orthogonal symplectic similarity transformation are returned in U);

Otherwise, (The transformation matrix is not required and U is not referenced).

3.6. Warning Indicator

None.

3.7. Error Indicator

IERR – INTEGER.

Unless the routine detects an error (see next section), IERR contains 0 on exit.

4. Warnings and Errors detected by the Routine

IERR = 1

On input, N < 1,
or LDA < N
or LDU < 1
or LDU < N and (COMPU = 'A' or 'a' or 'F' or 'f').

A.2 Example program

To transform the Hamiltonian matrix H to square-reduced form $\hat{H} = S^T H S$ where

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

and

$$A = \begin{bmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 9.0 \end{bmatrix}, \quad G = \begin{bmatrix} 1.0 & 1.0 & 1.0 \\ 1.0 & 2.0 & 2.0 \\ 1.0 & 2.0 & 3.0 \end{bmatrix}, \quad Q = \begin{bmatrix} 7.0 & 6.0 & 5.0 \\ 6.0 & 8.0 & 4.0 \\ 5.0 & 4.0 & 9.0 \end{bmatrix}.$$

Program Text

```
*     SQRED EXAMPLE PROGRAM TEXT.
*
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN = 5, NOUT = 6)
      INTEGER          NMAX
      PARAMETER        (NMAX = 20)
      INTEGER          LDA, LDU, LDGQ
      PARAMETER        (LDA = NMAX, LDGQ = NMAX*(NMAX+1)/2, LDU = NMAX)
      INTEGER          LRWORK
      PARAMETER        (LRWORK = 2*NMAX)
      DOUBLE PRECISION ZERO, ONE
      PARAMETER        (ZERO = 0.0D0, ONE = 1.0D0)
*     .. Local Scalars ..
      INTEGER          I, IERR, IJ, J, K, KJ, N, NSYM
      CHARACTER        COMPU
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA,NMAX), G(LDGQ), Q(LDGQ), U(LDU,NMAX),
     $                 RWORK(LRWORK)
*     .. External Subroutines ..
      EXTERNAL         DCOPY, DGEMM, DGEMV, DSCAL, DSPMV, SQRED
*     .. Executable Statements ..
*
      WRITE (NOUT,FMT=99999)
*     Skip the heading in the data file and read the data.
      READ (NIN,FMT='()')
      READ (NIN,FMT=*) N, COMPU
      IF (N .LE. 0 .OR. N .GT. NMAX) THEN
        WRITE (NOUT, FMT=99998) N
      ELSE
        NSYM = N*(N+1)/2
        READ (NIN,FMT=*) ((A(I,J),J=1,N),I=1,N)
        READ (NIN,FMT=*) (G(I),I=1,NSYM)
```

```
        READ (NIN,FMT=*) (Q(I),I=1,NSYM)
      ENDIF
*     .. square reduce by symplectic orthogonal similarity ..
      CALL SQRED(N, A, LDA, G, Q, U, LDU, RWORK, COMPU, IERR)
      IF (IERR.NE.0) THEN
        WRITE (NOUT,FMT=99997) IERR
      ELSE
*     .. show the square-reduced Hamiltonian ..
        WRITE (NOUT,FMT=99996)
        IJ = 1
        DO 20  I = 1, N
          CALL DCOPY(N, A(I,1), LDA, RWORK(1), 1)
          CALL DCOPY(N-I+1, G(IJ), 1, RWORK(N+I), 1)
          KJ = I
          DO 10 J = 1, I-1
            RWORK(N+J) = G(KJ)
            KJ         = KJ + N - J
10        CONTINUE
          WRITE (NOUT,FMT=99994) (RWORK(J),J=1,2*N)
          IJ = IJ + N - I + 1
20      CONTINUE
        IJ = 1
        DO 40  I = 1, N
          CALL DCOPY(N, A(1,I), 1, RWORK(N+1), 1)
          CALL DSCAL(N, -ONE, RWORK(N+1), 1)
          CALL DCOPY(N-I+1, Q(IJ), 1, RWORK(I), 1)
          KJ = I
          DO 30 J = 1, I-1
            RWORK(J) = Q(KJ)
            KJ       = KJ + N - J
30        CONTINUE
          WRITE (NOUT,FMT=99994) (RWORK(J),J=1,2*N)
          IJ = IJ + N - I + 1
40      CONTINUE
*     .. show the square of H ..
        WRITE (NOUT,FMT=99995)
        IJ = 1
        DO 80  I = 1, N
          CALL DGEMM('N', 'N', 1, N, N, ONE, A(I,1), LDA, A, LDA, ZERO,
     $               RWORK, 1)
          CALL DCOPY(N-I+1, G(IJ), 1, RWORK(N+I), 1)
          KJ = I
          DO 50 J = 1, I-1
            RWORK(N+J) = G(KJ)
            KJ         = KJ + N - J
50        CONTINUE
          CALL DSPMV('L', N, ONE, Q, RWORK(N+1), 1, ONE, RWORK(1), 1)
```

```
          CALL DSPMV('L', N, ONE, G, A(I,1), LDA, ZERO, RWORK(N+1), 1)
          CALL DGEMV('N', N, N-I+1, -ONE, A(1,I), LDA, G(IJ), 1, ONE,
     $               RWORK(N+1), 1)
          DO 70 J = 1, N
            KJ = I
            DO 60 K = 1, I-1
              RWORK(N+J) = RWORK(N+J) - A(J,K)*G(KJ)
              KJ         = KJ + N - K
60          CONTINUE
70        CONTINUE
          WRITE (NOUT,FMT=99994) (RWORK(J),J=1,2*N)
          IJ = IJ + N - I + 1
80      CONTINUE
        IJ = 1
        DO 120  I = 1, N
          CALL DGEMV('N', N, N, ONE, A, LDA, A(1,I), 1, ZERO,
     $               RWORK(N+1), 1)
          CALL DCOPY(N-I+1, Q(IJ), 1, RWORK(I), 1)
          KJ = I
          DO 90 J = 1, I-1
            RWORK(J) = Q(KJ)
            KJ       = KJ + N - J
90        CONTINUE
          CALL DSPMV('L', N, ONE, G, RWORK(1), 1, ONE, RWORK(N+1), 1)
          CALL DSPMV('L', N, ONE, Q, A(1,I), 1, ZERO, RWORK(1), 1)
          CALL DGEMV('T', N-I+1, N, -ONE, A(I,1), LDA, Q(IJ), 1, ONE,
     $               RWORK(1), 1)
          DO 110 J = 1, N
            KJ = I
            DO 100 K = 1, I-1
              RWORK(J) = RWORK(J) - A(K,J)*Q(KJ)
              KJ         = KJ + N - K
100         CONTINUE
110       CONTINUE
          WRITE (NOUT,FMT=99994) (RWORK(J),J=1,2*N)
          IJ = IJ + N - I + 1
120     CONTINUE
      END IF
      STOP
*
99999 FORMAT (' SQRED EXAMPLE PROGRAM RESULTS',/1X)
99998 FORMAT (/' N is out of range.',/' N = ',I5)
99997 FORMAT (' IERR on exit from SQRED = ',I2)
99996 FORMAT (/' The square-reduced Hamiltonian is ')
99995 FORMAT (/' The square of the square-reduced Hamiltonian is ')
99994 FORMAT (1X,8(F10.4))
      END
```

Program Data

```
SQRED EXAMPLE PROGRAM DATA
3 N
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0
1.0 1.0 1.0 2.0 2.0 3.0
7.0 6.0 5.0 8.0 4.0 9.0
```

Program Results

```
 SQRED EXAMPLE PROGRAM RESULTS


 The square-reduced Hamiltonian is
     1.0000    3.3485     .3436    1.0000    1.9126    -.1072
     6.7566   11.0750    -.3014    1.9126    8.4479   -1.0790
     2.3478    1.6899   -2.3868    -.1072   -1.0790   -2.9871
     7.0000    8.6275    -.6352   -1.0000   -6.7566   -2.3478
     8.6275   16.2238    -.1403   -3.3485  -11.0750   -1.6899
     -.6352    -.1403    1.2371    -.3436     .3014    2.3868

 The square of the square-reduced Hamiltonian is
    48.0000   80.6858   -2.5217     .0000    1.8590  -10.5824
   167.8362  298.4815   -4.0310   -1.8590     .0000  -33.1160
      .0000    4.5325    2.5185   10.5824   33.1160     .0000
      .0000     .0000     .0000   48.0000  167.8362     .0000
      .0000     .0000     .0000   80.6858  298.4815    4.5325
      .0000     .0000     .0000   -2.5217   -4.0310    2.5185
```

B The Fortran 77 Subroutine HAMEV

This section describes the subroutine HAMEV and gives an example program showing how to use HAMEV. The description and example program text follow the SLICOT subroutine documentation standard [25].

B.1 Subroutine description

1. Purpose

To approximate the eigenvalues of a Hamiltonian matrix

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \tag{20}$$

by Van Loan's square reduced method [37]. Here, A, G, and Q are N×N matrices where G and Q are symmetric. The Hamiltonian matrix H is transformed to square-reduced form via a similarity transformation

$$\hat{H} := \frac{1}{\tau} U^T D^{-1} H D U =: \begin{bmatrix} \hat{A} & \hat{G} \\ \hat{Q} & -\hat{A}^T \end{bmatrix} \tag{21}$$

where U is orthogonal and symplectic, $D = \mathrm{diag}(D_0, D_0^{-1})$ is a symplectic diagonal matrix, and τ is a real scalar. The eigenvalues of H are computed as the positive and negative square roots of the eigenvalues of $\hat{A}^2 + \hat{G}\hat{Q}$.

2. Specification

```
 SUBROUTINE       HAMEV(N, A, LDA, GP, QP, U, LDU, SCALE, NPI, WR, WI,
1                       RWORK, TOL, COMPU, JOBEV, JOBSCL, ORDER, IERR)
 INTEGER          N, LDA, LDU, NPI, IERR
 DOUBLE PRECISION A(LDA,N), GP(N*(N+1)/2), QP(N*(N+1)/2), U(LDU,2*N),
1                 SCALE(N+1), WR(N), WI(N), RWORK(N*(N+1))
 CHARACTER        COMPU, JOBEV, JOBSCL, ORDER
```

3. Argument List

3.1. Arguments In

N – INTEGER.

The order of the matrices A, G, and Q.
N $\geq$ 1.

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N×N part of this array contains the upper left block A of the Hamiltonian matrix H in (20).
Note that this array is overwritten.

LDA – INTEGER.

The leading dimension of array A as declared in the calling program.
LDA $\geq$ N.

GP, QP – DOUBLE PRECISION arrays of DIMENSION at least N*(N+1)/2.

Array GP contains the upper right block G, array QP the lower left block Q of the Hamiltonian matrix H in (20) in lower packed storage mode, i.e., the lower triangles of the symmetric matrices are stored by columns.

Note that these arrays are overwritten.

U – DOUBLE PRECISION array of DIMENSION (LDU,*udim*).

If COMPU = 'A' or 'a' or 'F' or 'f', then *udim* $\geq$ 2*N. Otherwise, *udim* $\geq$ 1.
If COMPU = 'A' or 'a', the leading N$\times$2N part of this array **must** contain the first N rows of an orthogonal symplectic matrix.

Note that this array is overwritten if COMPU = 'A' or 'a' or 'F' or 'f'.

LDU – INTEGER.

The leading dimension of array U as declared in the calling program.
If COMPU = 'A' or 'a' or 'F' or 'f', then LDU $\geq$ N. Otherwise, LDU $\geq$ 1.

3.2. Arguments Out

A – DOUBLE PRECISION array of DIMENSION (LDA,N).

The leading N$\times$N part of this array contains the upper left block $\hat{A}$ of the square-reduced Hamiltonian matrix $\hat{H}$ of (21).

GP, QP – DOUBLE PRECISION arrays of DIMENSION at least N*(N+1)/2.

Array GP contains the upper right block $\hat{G}$ and array QP the lower left block $\hat{Q}$ of the square-reduced Hamiltonian matrix of (21) in lower packed storage mode, i.e., the lower triangles of the symmetric matrices are stored by columns.

U – DOUBLE PRECISION array of DIMENSION (LDU,*udim*).

If COMPU = 'A' or 'a' or 'F' or 'f', then *udim* $\geq$ 2*N. Otherwise, *udim* $\geq$ 1.
If COMPU = 'F' or 'f', then array U contains the first N rows of the orthogonal symplectic matrix of (21).
IF COMPU = 'A' or 'a', then array U contains the first N rows of the product of an orthogonal symplectic input matrix and the orthogonal symplectic matrix U of (21).
Otherwise, U is not referenced.

SCALE – DOUBLE PRECISION array of DIMENSION at least *scldim*.

If JOBSCL = 'A' or 'a', then *scldim* $\geq$ N+1. Otherwise, *scldim* $\geq$ 1.
On output, SCALE contains all information to form the symplectic diagonal matrix $D = \mathrm{diag}(D_0, D_0^{-1})$ of (21).
If JOBSCL = 'A' or 'a', then

$$D_0 = \frac{1}{\sqrt{\mathrm{SCALE(N+1)}}} \times \mathrm{diag}(\mathrm{SCALE}(1), \ldots, \mathrm{SCALE(N)})$$

and τ = 1.0. Otherwise, $D_0 = \sqrt{\mathrm{SCALE}(1)} \times I_{\mathrm{N}}$ where I_{N} denotes the N$\times$N identity matrix, and τ = SCALE(1).

NPI – INTEGER.

The number of returned eigenvalues having real part zero with respect to the relative tolerance TOL (see Section **Tolerances**). This number is only computed if ORDER = 'O' or 'o'. Otherwise, NPI is not referenced.

WR, WI – DOUBLE PRECISION arrays of DIMENSION at least *nev*.

If JOBEV = 'S' or 's' or 'U' or 'u', then *nev* = N, i.e., only stable ('S', 's') or unstable ('U', 'u') eigenvalues are returned. Otherwise, *nev* = 2*N, and all the eigenvalues of the Hamiltonian matrix H of (20) are returned.

WR contains the real parts, WI the imaginary parts of the required eigenvalues. Complex conjugate pairs of eigenvalues appear consecutively if their real part is not zero. If all the eigenvalues are required, then the first N components of WR, WI contain the stable eigenvalues, followed by the unstable ones, i.e.,
(WR(N+I),WI(N+I)) = −(WR(I),WI(I)), $1 \leq$ I $\leq$N.

As a consequence, if H has a purely imaginary eigenvalue of odd multiplicity, say (WR(I),WI(I)), then its conjugate complex partner is −(WR(N+I),WI(N+I)).

3.3. Work space

RWORK – DOUBLE PRECISION array of DIMENSION at least 2*N.

3.4. Tolerances

TOL – DOUBLE PRECISION.

A tolerance used for deciding if a computed eigenvalue λ is considered to be purely imaginary.

$|\mathrm{Re}(\lambda)| \leq \mathrm{TOL} \times |\lambda| \Longrightarrow \lambda$ is purely imaginary.

If on input, TOL is less than zero, then TOL is set to the default value $10.0 \times \sqrt{\varepsilon}$ (here, ε denotes the machine precision). This is inspired by the error analysis for eigenvalues computed by SQRED (see [12] and [37]).

TOL is only used if ORDER = 'O' or 'o' (see next section). Otherwise, TOL is not referenced.

3.5. Mode Parameters

COMPU – CHARACTER.

Indicates whether the orthogonal symplectic matrix U of (21) is returned or accumulated into an input matrix or if the transformation matrix is not required.

COMPU = 'A' or 'a', (The orthogonal symplectic similarity transformations are accumulated in U, i.e., array U contains the first N rows of the product of an orthogonal symplectic input matrix and the similarity transformation matrix of (21). On input, array U **must** contain the first N rows of an orthogonal symplectic 2N×2N matrix);

COMPU = 'F' or 'f', (The matrix U of (21) is formed, i.e., the first N rows of the orthogonal symplectic similarity transformation are returned in U);

Otherwise, (The transformation matrix is not required and S is not referenced).

JOBEV – CHARACTER.

Indicates whether the stable, unstable, or all the eigenvalues are required.

JOBEV = 'S' or 's', (Only the stable eigenvalues are required, i.e., N eigenvalues with nonpositive real parts are returned);

JOBEV = 'U' or 'u', (Only the unstable eigenvalues are required, i.e., N eigenvalues with nonnegative real parts are returned);
Otherwise, (All the eigenvalues are required).

JOBSCL – CHARACTER.

Indicates which scaling strategy is used as follows, for details see Section 3.1.

JOBSCL = 'A' or 'a', (Symplectic scaling);

JOBSCL = 'B' or 'b', (Norm scaling);

JOBSCL = 'N' or 'n', (No scaling).

For any input value different from 'N' or 'n', the rows and columns of $\hat{A}^2 + \hat{G}\hat{Q}$ are equilibrated in norm as far as possible using LAPACK subroutine DGEBAL [4] before computing its eigenvalues.

ORDER – CHARACTER.

Indicates whether the computed eigenvalues are ordered.

ORDER = 'O' or 'o', (If purely imaginary eigenvalues are detected with respect to TOL, (see Section **Tolerances**), they are placed at the end of WR, WI (or the end of each half of WR, WI if all the eigenvalues are returned). If possible, they are grouped as complex conjugate pairs).

Otherwise, the eigenvalues are returned in the order they are computed.

3.6. Warning Indicator

None.

3.7. Error Indicator

IERR – INTEGER.

Unless the routine detects an error (see next section), IERR contains 0 on exit.

4. Warnings and Errors detected by the Routine

IERR = 1

On input, N < 1,
or LDA < N
or LDU < 1
or LDU < N and (COMPU = 'A' or 'a' or 'F' or 'f').

IERR > 10

On input to DGEBAL, the (IERR−10)th argument had an illegal value.

IERR = −J

If the limit of 30∗N iterations is exhausted while the J–th eigenvalue is being sought, see LAPACK subroutines DLAHQR or DHSEQR.

B.2 Example program

To compute the eigenvalues of

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

where

$$A = \begin{bmatrix} 2.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 2.0 \\ 0.0 & -1.0 & 3.0 \end{bmatrix}, \quad G = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 2.0 & 3.0 \\ 0.0 & 3.0 & 4.0 \end{bmatrix}, \quad \text{and} \quad Q = \begin{bmatrix} -2.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{bmatrix}.$$

Program Text

```
*     HAMEV EXAMPLE PROGRAM TEXT.
*
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN = 5, NOUT = 6)
      INTEGER          NMAX
      PARAMETER        (NMAX = 20)
      INTEGER          LDA, LDGQ, LDU
      PARAMETER        (LDA = NMAX, LDGQ = NMAX*(NMAX+1)/2, LDU = NMAX)
      INTEGER          LRWORK
      PARAMETER        (LRWORK = NMAX*(NMAX+1))
*     .. Local Scalars ..
      DOUBLE PRECISION TOL
      INTEGER          I, IERR, J, N, NPI, NSYM
      CHARACTER        COMPU, JOBEV, JOBSCL, ORDER
*     .. Local Arrays ..
      DOUBLE PRECISION A(LDA, NMAX), G(LDGQ), Q(LDGQ), SCALE(NMAX+1),
     $                 U(LDU,2*NMAX), WR(2*NMAX), WI(2*NMAX),
     $                 RWORK(LRWORK)
*     .. External Functions
      LOGICAL          LSAME
      EXTERNAL         LSAME
*     .. External Subroutines ..
      EXTERNAL         HAMEV
*     .. Executable Statements ..
*
      WRITE (NOUT,FMT=99999)
*     Skip the heading in the data file and read the data.
      READ (NIN,FMT='()')
      READ (NIN,FMT=*) N, TOL, COMPU, JOBEV, JOBSCL, ORDER
      IF (N .LE. 0 .OR. N .GT. NMAX) THEN
        WRITE (NOUT,FMT=99998) N
      ELSE
        NSYM = N*(N+1)/2
        READ (NIN,FMT=*) ((A(I,J),J=1,N),I=1,N)
```

```
        READ (NIN,FMT=*) (G(I),I=1,NSYM)
        READ (NIN,FMT=*) (Q(I),I=1,NSYM)
        IF (LSAME(COMPU,'A'))  READ (NIN,FMT=*) ((U(I,J),J=1,2*N),I=1,N)
*       Compute the eigenvalues.
        CALL HAMEV(N, A, LDA, G, Q, U, LDU, SCALE, NPI, WR, WI, RWORK,
     $             TOL, COMPU, JOBEV, JOBSCL, ORDER, IERR)
*
        IF (IERR .NE. 0) THEN
          WRITE (NOUT,FMT=99997) IERR
        ELSE
*       Show number of purely imaginary eigenvalues.
          IF (LSAME(ORDER,'O')) THEN
            WRITE (NOUT,FMT=99996) TOL
            WRITE (NOUT,FMT=99995) NPI
          END IF
*       Show the computed eigenvalues.
          WRITE (NOUT,FMT=99994)
          DO 10  I = 1, N
            WRITE (NOUT,FMT=99993) WR(I), ' + (', WI(I), ')i'
10        CONTINUE
          IF (LSAME(JOBEV,'A')) THEN
            DO 20  I = 1, N
              WRITE (NOUT,FMT=99993) WR(N+I), ' + (', WI(N+I), ')i'
20          CONTINUE
          END IF
        END IF
      END IF
      STOP
*
99999 FORMAT (' HAMEV EXAMPLE PROGRAM RESULTS',/1X)
99998 FORMAT (/' N is out of range.',/' N = ',I5)
99997 FORMAT (' IERR on exit from HAMEV = ',I2)
99996 FORMAT (/' Relative tolerance TOL for small imaginary parts :',
     $        G10.3)
99995 FORMAT (' Number of purely imaginary eigenvalues w.r.t. TOL:',I3)
99994 FORMAT (/' The eigenvalues are ')
99993 FORMAT (1X,F8.4,A,F8.4,A)
      END
```

Program Data

```
HAMEV EXAMPLE PROGRAM DATA
 3 .1D-11 'N' 'A' 'H' 'O'
  2.0  0.0  0.0
  0.0  1.0  2.0
  0.0 -1.0  3.0
  1.0  0.0  0.0  2.0  3.0  4.0
 -2.0  0.0  0.0  0.0  0.0  0.0
```

Program Results

```
 HAMEV EXAMPLE PROGRAM RESULTS


 Relative tolerance TOL for small imaginary parts :  .100E-11
 Number of purely imaginary eigenvalues w.r.t. TOL:  0

 The eigenvalues are
  -1.4142 + (   .0000)i
  -2.0000 + ( -1.0000)i
  -2.0000 + (  1.0000)i
   1.4142 + (   .0000)i
   2.0000 + (  1.0000)i
   2.0000 + ( -1.0000)i
```

C How to Obtain the Software

The codes corresponding to this paper may be obtained via anonymous ftp at TU Chemnitz-Zwickau. Proceed as follows.

> > `ftp ftp.tu-chemnitz.de`
> > Name: `anonymous`
> > Password: *your complete e-mail address*
> > `cd /pub/Local/mathematik/Benner`
> > `binary`

Observe the capital "L" in `Local` !
You can obtain the complete set of source codes (*hamev.f, sqred.f, croot.f, hamgiv.f, syref.f*) by

> > `get hamev.tar`*.gz*

or just the codes for the reduction to square-reduced form (*sqred.f, hamgiv.f, syref.f*) by

> > `get sqred.tar`*.gz*

where the suffix *.gz* is optional. Without this suffix, you obtain the tar-files `hamev.tar`, `sqred.tar` whereas using the suffix *.gz*, the tar files are compressed using `gzip`.
Both tar files contain the source codes, the example programs given in Appendices B.2 and A.2 together with the data and resulting output files. Also included are introductory *README* files and validation programs.

After exiting ftp, extracting the files (after possibly uncompressing using `gunzip`) is achieved by

> > `tar xf hamev.tar`

or

> > `tar xf sqred.tar`

In both cases, a directory is created containing all required files. For `hamev.tar`, this directory is called *hamev* and for `sqred.tar`, it will be *sqred*.
If you are using an MS DOS or MS WINDOWS environment and a tar command is not available on your sytem, you can obtain `tar.exe` at the same ftp site from directory

> > `/pub/tex/tools/tar/msdos`

Fortran 77 files for compressing a symmetric or triangular matrix to packed storage mode and for uncompressing an array in packed storage mode are also available at the same location by

> > `get sysp.tar`*.gz*

If any problems occur in obtaining or running the codes, please send an e-mail message to

> > `benner@mathematik.tu-chemnitz.de`

References

[1] G. Ammar and P. Benner, *OSMARE: A Fortran 77 implementation of the orthogonal symplectic multishift method for the continuous-time algebraic Riccati equation*, preprint, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1996. 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., 1994.

[5] 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. Report SPC 95_22, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995.

[6] P. Benner, V. Mehrmann, and H. Xu, *A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils*, Tech. Report SFB393/96-05, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1996.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

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

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

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

[15] 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.

[16] 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.

[17] J. Doyle, K. Glover, P. Khargonekar, and B. Francis, *State-space solutions to standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ control problems*, IEEE Trans. Automat. Control, AC-34 (1989), pp. 831–847.

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

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

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

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

[22] A. Laub, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913–921. (see also *Proc. 1978 CDC (Jan. 1979)*, pp. 60-65).

[23] 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.

[24] V. Mehrmann and E. Tan, *Defect correction methods for the solution of algebraic Riccati equations*, IEEE Trans. Automat. Control, 33 (1988), pp. 695–698.

[25] Numerical Algorithms Group, *Implementation and documentation standards for the subroutine library in control and systems theory SLICOT*, Publication NP2032, Numerical Algorithm Group, Eindhoven/Oxford, 1990.

[26] E. Osborne, *On pre-conditioning of matrices*, J. ACM, 7 (1960), pp. 338–345.

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

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

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

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

[31] 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.

[32] 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.

[33] 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.

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

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

[36] P. Van Dooren, *Fast computation of the real and complex stability radius.* Talk given at IMA Conference on *Linear Algebra and Applications*, Manchster, UK, July 1995.

[37] C. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 16 (1984), pp. 233–251.

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

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

[40] 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. 127–146.

[41] K. Zhou, K. Glover, B. Bodenheimer, and J. Doyle, *Mixed $\mathcal{H}_2$ and $\mathcal{H}_\infty$ performance objectives I: Robust performance analysis*, IEEE Trans. Automat. Control, 39 (1994), pp. 1564–1574.