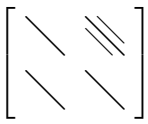
A DETAILED DERIVATION OF THE PARAMETERIZED SR ALGORITHM AND THE SYMPLECTIC LANCZOS METHOD FOR HAMILTONIAN MATRICES*

H. FAßBENDER†

Abstract. The implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem is reconsidered here; restarts like the implicit restart als Sorensen and ala Stewart will be discussed. The Lanczos vectors are constructed to form a symplectic basis. The inherent numerical difficulties of the symplectic Lanczos method are addressed by inexpensive implicit restarts. The heart of the implicitly restarted symplectic Lanczos method for Hamiltonian matrices consists of the SR algorithm, a structure-preserving algorithm for computing the spectrum of Hamiltonian matrices. The symplectic Lanczos method projects the large, sparse $2n \times 2n$ Hamiltonian matrix H onto a small, dense $2k \times 2k$ Hamiltonian J-Hessenberg matrix $\widetilde{H}, k \ll n$. This $2k \times 2k$ Hamiltonian matrix is uniquely determined by $4k-1$ parameters. Using these $4k-1$ parameters, we show how one step of the SR algorithm can be carried out in $\mathcal{O}(k)$ arithmetic operations (compared to $\mathcal{O}(k^3)$ arithmetic operations when working on the actual Hamiltonian matrix). Moreover, the Hamiltonian structure, which will be destroyed in the numerical process due to roundoff errors when working with a Hamiltonian matrix, will be forced by working just with the parameters. As in the context of the implicitly restarted symplectic Lanczos method the usual assumption, that the Hamiltonian eigenproblem to be solved is stable, does not hold, the case of purely imaginary eigenvalues in the SR algorithm is treated.

Key words. Hamiltonian Matrix; Eigenvalue Problem; SR Algorithm; Symplectic Lanczos Algorithm.
AMS(MOS) subject classifications. 65F15.

1. Introduction. Renewed interest [94, 145, 19] in the implicitly restarted symplectic Lanczos method for computing a few eigenvalues of a large, sparse Hamiltonian matrix [17] led us to reconsider that algorithm. It projects the large, sparse $2n \times 2n$ Hamiltonian matrix H onto a small, dense $2k \times 2k$ Hamiltonian J-Hessenberg matrix $\widetilde{H}$, $k \ll n$. This matrix $\widetilde{H}$ is of the form



that is, due to the Hamiltonian structure, it can be represented by $4k-1$ parameters instead of the usual k^2 matrix entries. As observed in [38], the SR algorithm preserves the Hamiltonian J-Hessenberg form. A standard implementation of the SR algorithm will require $\mathcal{O}(k^3)$ flops in each iteration step. As pointed out in [38], using the $4k-1$ parameters one step of the SR algorithm for H can be carried out in $\mathcal{O}(k)$ flops. Usually, this algorithm s implemented such that it works on the Hamiltonian J-Hessenberg matrix itself, working in a narrow band around the diagonals of the J-Hessenberg form. But, if no extra care is taken, the Hamiltonian structure will be destroyed in the numerical process due to roundoff errors when working with a Hamiltonian (J-Hessenberg) matrix.

In [38], the algorithm is discussed only for the case that the Hamiltonian matrix is stable, that is, it has no eigenvalues on the imaginary axis. While this a reasonable assumption in a number of applications, in the context of a restarted symplectic

*WORK IN PROGRESS, VERSION NOVEMBER 6, 2006

†Technische Universität Braunschweig, Institut *Computational Mathematics*, 38106 Braunschweig, Germany, email: h.fassbender@tu-bs.de

Lanczos method in which small eigenproblems of Hamiltonian J-Hessenberg form have to be solved, this cannot be assumed for the small eigenproblems even if the original problem is stable.

Therefore, in this paper, we will develop an implementation of the SR algorithm that cures the drawbacks of the existing SR algorithm for Hamiltonian matrices. The implementation developed here can deal with eigenvalues on the imaginary axis. Moreover, by working only with the $4k-1$ parameters describing the Hamiltonian J-Hessenberg matrix, the implementation of one step of the SR algorithm can be carried out in $\mathcal{O}(k)$ flops. The Hamiltonian structure is forced in each step; roundoff errors cannot destroy the Hamiltonian structure. Our goal will be to derive a description of an implicit SR step such that the parameters which represent the resulting matrix are computed directly from the original ones without ever forming the actual Hamiltonian J-Hessenberg matrix or the bulge which is chased in the SR step. Numerical experiments indicate that this extra care might make a positive difference in the accuracy of the computed results.

Unfortunately, the presentation of the implicitly restarted symplectic Lanczos algorithm in [17] considered only a single shift implicit restart and the discussion of its properties is incorrect. Therefore, the implicitly restarted symplectic Lanczos algorithm itself is reconsidered here in detail. Moreover, an implicit restart like the Krylov-Schur restart proposed by Stewart [131, 130] will be discussed.

A Hamiltonian matrix $H \in \mathbb{R}^{2n\times 2n}$ has the form

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix}, \qquad G = G^T, \quad Q = Q^T, \tag{1.1}$$

where A, G and Q are real $n \times n$ matrices. A number of applications from control theory and related areas lead to eigenvalue problems

- stability radius and H_∞ norm computation [43, 31]
- linear quadratic optimal control problems and the solution of continuous-time algebraic Riccati equations [12, 98, 123]
- H_∞ control [14, 15]
- passivity preserving model reduction [5, 129, 19]
- quadratic eigenvalue problems [100, 135]
- computation of pseudo-spectra [40]

An ubiquitous matrix when dealing with Hamiltonian eigenvalue problems is the skew-symmetric matrix

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}, \tag{1.2}$$

where I denotes the $n \times n$ identity matrix. By straightforward algebraic manipulation one can show that a Hamiltonian matrix H is equivalently defined by the property

$$HJ = (HJ)^T. \tag{1.3}$$

Any matrix $S \in \mathbb{R}^{2n\times 2n}$ satisfying

$$S^T JS = SJS^T = J$$

is called symplectic, and since

$$(S^{-1}HS)J = S^{-1}HJS^{-T} = S^{-1}J^T H^T S^{-T} = [(S^{-1}HS)J]^T, \tag{1.4}$$

we see that symplectic similarity transformations preserve Hamiltonian structure. There are relevant cases, however, where both H and $S^{-1}HS$ are Hamiltonian, but S is not a symplectic matrix [60].

One of the most remarkable properties of a Hamiltonian matrix is that its eigenvalues always occur in pairs $\{\lambda, -\lambda\}$ if λ is real or purely imaginary, or in quadruples $\{\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}\}$ otherwise. Hence, the spectrum of any Hamiltonian matrix is symmetric with respect to the real and imaginary axis. Numerical methods that take this structure into account are capable of preserving the eigenvalue pairings despite the presence of roundoff errors. Figure 1.1, which displays the eigenvalues of a Hamiltonian matrix stemming from a discretized Maxwell equation (obtained via linearizing the quadratic eigenvalue) [122] illustrates this fact. The exact eigenvalues are represented as grey dots in each plot; one can clearly see the eigenvalue pairing $\{\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}\}$ for complex eigenvalues with nonzero real part and $\{\lambda, -\lambda\}$ for real and purely imaginary eigenvalues. Eigenvalue approximations (denoted by black crosses) have been computed with two different types of Arnoldi methods: in the plot on the left hand side the eigenvalue approximations were obtained from a few iterations of the standard Arnoldi method [65] while the ones in the plot on the right hand side were obtained from the same number of iteration of an Arnoldi method that takes Hamiltonian structures into account [100]. The latter method clearly produces pairs of eigenvalue approximations. Besides the preservation of such eigenvalue symmetries, there are several other benefits to be gained from using structure-preserving algorithms in place of general-purpose algorithms for computing eigenvalues. These benefits include reduced computational time and improved eigenvalue/-eigenvector accuracy.

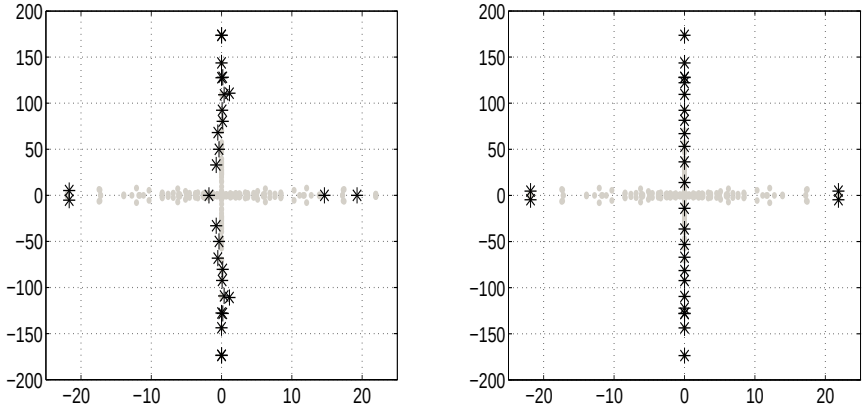


FIG. 1.1. *Eigenvalues ('·') and approximate eigenvalues ('⋆') computed by a standard Arnoldi method (left picture) and by a structure-preserving Arnoldi method (right picture).*

Hence, in order to develop fast and efficient numerical methods for the Hamiltonian eigenproblem one should make use of the rich mathematical structure of the problem. This has been successfully done for symmetric/Hermitian and orthogonal/unitary eigenproblems. E.g., for the symmetric eigenproblem, one of the nowadays standard approaches involves first the reduction of the symmetric matrix to symmetric tridiagonal form followed by a sequence of implicit QR steps which preserve this symmetric tridiagonal form, see, e.g., [65]. Such structure preserving methods are

desirable as important properties of the original problem are preserved during the actual computations and are not destroyed by rounding errors. Moreover, in general, such methods allow for faster computations than general-purpose methods. For the symmetric eigenproblem, e.g., applying implicit QR steps to the full symmetric matrix requires $\mathcal{O}(n^3)$ arithmetic operations per step, while applying an implicit QR step to the similar symmetric tridiagonal matrix requires only $\mathcal{O}(n)$ arithmetic operations, where n is the order of the matrix. If the matrix under consideration is large and sparse, the QR method might not be a suitable tool for computing the eigeninformation. In that case, usually the Lanczos method [82, 65], a technique especially tuned to solve large, sparse symmetric eigenproblems should be used.

The eigenvalues and invariant subspaces of Hamiltonian matrices H may be computed by the QR algorithm [65]. But the QR method cannot take advantage of the Hamiltonian structure of H, it will treat H like any arbitrary $2n \times 2n$ matrix. The computed eigenvalues will in general not come in quadruple $\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}$, although the exact eigenvalues have this property. Even worse, small perturbations may cause eigenvalues close to the imaginary axis to cross the axis such that the number of true and computed eigenvalues in the right half plane may differ.

To preserve the Hamiltonian structure of H, we would have to employ similarity transformations with symplectic matrices instead of the transformations with the usual unitary matrices in the QR algorithm. Under certain conditions a Hamiltonian matrix H may be reduced to Hamiltonian Hessenberg form

$$U^T H U = \begin{bmatrix} \blacksquare & \square \\ * & \blacksquare \end{bmatrix}$$

using a symplectic and orthogonal transformation matrix U. Byers [41] derived a simple method for reducing H to such a form under the assumption that one of the off-diagonal blocks G or Q in H has tiny rank, i.e., rank 1, 2 or at most 3. This form stays invariant under a QR like iteration which uses only symplectic and orthogonal transformations. However, the computation of the initial unreduced Hamiltonian Hessenberg form is not always possible. As shown in [3] the components of the first column of U must satisfy a system of n quadratic equations in $2n$ unknowns. Consequently, such a reduction is not always possible. Hence, more general QR like methods have to be considered in order to derive a structure-preserving QR like eigenvalue method for the Hamiltonian eigenproblem.

Paige and Van Loan introduced in [108] the Hamiltonian Schur form of a (complex) Hamiltonian matrix. For any Hamiltonian matrix

$$M = \begin{bmatrix} A & N \\ K & -A^H \end{bmatrix}, \qquad N^H = N, K^H = K, A, N, K \in \mathbb{C}^{n\times n}$$

whose eigenvalues have nonzero real part, there exists a unitary and symplectic matrix

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ -Q_{12} & Q_{11} \end{bmatrix}, \qquad Q_{11}, Q_{12} \in \mathbb{C}^{n\times n}$$

such that

$$Q^H M Q = \begin{bmatrix} T & R \\ 0 & -T^H \end{bmatrix}, \qquad T, R \in \mathbb{C}^{n\times n}$$

where T is upper triangular and $R^H = R$. Q can be chosen so that the eigenvalues of T are in the left half plane. Lin and Ho [91] extended this result to the case that M has eigenvalues on the imaginary axis. See also [81, 92]. In case H is real, a real orthogonal and symplectic matrix Q can be found such that

$$Q^T M Q = \begin{bmatrix} T & R \\ 0 & -T^T \end{bmatrix}, \qquad T, R \in \mathbb{R}^{n \times n}$$

where T is upper quasi-triangular and $R^T = R$. Q can be chosen so that the eigenvalues of T are in the left half plane. Unfortunately, the numerical computation of the Hamiltonian Schur form via strongly backward stable $\mathcal{O}(n^3)$ method has been an open problem since its introduction. Only in special cases a satisfactory solution has been obtained [42]. If the Hamiltonian QR algorithm has successfully computed a Hamiltonian Schur decomposition, then the first n columns of the orthogonal symplectic matrix Q span an isotropic subspace belonging to the eigenvalues of T. Many applications require the stable invariant subspace, for this purpose the Schur decomposition has to be reordered so that T contains all eigenvalues with negative real part. See [41] for details. A perturbation analysis for the Hamiltonian Schur form can be found in [75].

Jacobi-like methods for computing the Hamiltonian Schur form are proposed in [44, 37]. In [56] Jacobi-like algorithm for solving the complete eigenproblem for Hamiltonian (and skew-Hamiltonian) matrices that are also symmetric or skew-symmetric are developed. These methods employ a sequence of easy to determine unitary symplectic similarity transformations. In addition to preserving structure, these algorithms are inherently parallelizable, and numerically stable. The stability in particular of the latter algorithms has been investigated in [134].

In [136] Van Loan suggests the squared-reduced method for computing all eigenvalues of a Hamiltonian matrix. In this method, a real Hamiltonian matrix H is first squared $H^2 = N$, then a symplectic orthogonal matrix Q is constructed such that

$$Q^T N Q = \begin{bmatrix} X & R \\ 0 & X^T \end{bmatrix}$$

where X is upper Hessenberg. The QR algorithm is used to compute the eigenvalues $\mu_1, \ldots, \mu_n$ of X. The eigenvalues λ_j of H can be obtained by taking the square-roots located in the left-half plane of μ_j: $\lambda_j = \sqrt{\mu_j}, \lambda_{n+j} = -\lambda_j$ for $j = 1, \ldots, n$. Unfortunately a loss of half of the possible accuracy in the eigenvalues is possible when using this algorithm [136].

Based on the ideas of Paige and Van Loan [108] and Van Loan [136], Ammar, Benner and Mehrmann [1] proposed the multishift algorithm for computing an invariant subspace of dimension n of H. First, all eigenvalues of H are computed via the square-reduced method [136], next the n desired eigenvalues $\lambda_1, \ldots, \lambda_n$ are chosen, $x = (H - \lambda_1 I) \ldots (H - \lambda_n I) e_1$ is formed and a symplectic orthogonal matrix Q_1 is computed such that $Q_1^T x = \alpha_1 e_1$. Next $Q_1^T H Q_1$ is reduced to the Hamiltonian Schur form [108]. Essentially, the desired invariant subspace can be read off.

In [27], Benner, Mehrmann and Xu propose a numerically backward stable method to compute the eigenvalues (but not the invariant subspaces) of real Hamiltonian matrices using an approach via non-similarity transformations which exploits that one can compute the real skew-Hamiltonian matrix H^2 without forming the square and which makes use of a symplectic URV decomposition. Its drawback is that it does not take full advantage of the structure of H. It is not yet clear whether the

method is strongly backward stable. Based on this algorithm, in [26], the authors employ a relationship between the eigenvalues and invariant subspaces of H and the extended matrix

$$\begin{bmatrix} 0 & H \\ H & 0 \end{bmatrix}$$

to develop a backward stable, structure preserving $\mathcal{O}(n^3)$ method for computing all eigenvalues and invariant subspaces of H. The extension of these algorithms to the complex case is considered in [28]. Implementations of the algorithms in [27] and [26] are freely available from the HAPACK package[1][23]. Using the ideas of [26, 27], Chu, Liu and Mehrmann suggest in [48] a numerically strongly backward stable $\mathcal{O}(n^3)$ method for computing the Hamiltonian Schur form of a Hamiltonian matrix that has no purely imaginary eigenvalues.

Algorithms based on symplectic but non-orthogonal transformations include the SR algorithm (which will be studied in detail in the following sections) [38, 98] and related methods [39, 115]. The SR algorithm is applicable to real matrices of even dimensions $2n \times 2n$. Almost every matrix $A \in \mathbb{R}^{2n\times 2n}$ can be decomposed into a product $A = SR$ where S is symplectic and R is J-triangular [53], that is,

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

where all submatrices $R_{ij} \in \mathbb{R}^{n\times n}$ are upper triangular, and R_{21} is strictly upper triangular (if one performs a perfect shuffle of the rows and columns of a J-triangular matrix, one gets an upper triangular matrix). The SR decomposition is essentially unique (that is, the factorization is unique upto symplectic factors of the form

$$\begin{bmatrix} C^{-1} & F \\ 0 & C \end{bmatrix},$$

where C and F are $n\times n$ diagonal matrices). The SR algorithm is an QR-like iterative algorithm that performs an SR decomposition at each iteration. If H_i is the current iterate, then a spectral transformation function q is chosen and the SR decomposition of $q(H_i)$ is formed, if possible:

$$q(H_i) = SR.$$

Then the symplectic factor S is used to perform a similarity transformation on H to yield the next iterate,

$$H_{i+1} = S^{-1}H_iS.$$

As only symplectic similarity transformations are performed, the SR algorithm preserves the Hamiltonian structure.

Any Hamiltonian matrix can be reduced to Hamiltonian J-Hessenberg form [38]

$$\begin{bmatrix} \diagdown & \diagdown\!\!\diagdown \\ \diagdown & \diagdown \end{bmatrix},$$

[1] see the HAPACK homepage http://www.tu-chemnitz.de/mathematik/hapack/

where each block is an $n \times n$ matrix. Due to the Hamiltonian structure, the $(1,1)$ and the $(2,2)$ block are identical, while the $(1,2)$ block is symmetric. Hence any Hamiltonian matrix can be represented by $4n-1$ parameters. As observed in [38], the SR algorithm preserves the Hamiltonian J-Hessenberg form. A standard implementation of the SR algorithm will require $\mathcal{O}(n^3)$ flops in each iteration step. Using the $4n-1$ parameters we show in the following sections how one step of the SR algorithm for H can be carried out in $\mathcal{O}(n)$ flops. Moreover, the Hamiltonian structure which will be destroyed in the numerical process due to roundoff errors when working with a Hamiltonian (J-Hessenberg) matrix, will be forced by working just with the parameters. The general convergence theory for GR methods developed by Elsner and Watkins in [142] implies that the SR algorithm for Hamiltonian matrices is typically cubically convergent. A connection between the SR algorithm on a $2n \times 2n$ Hamiltonian J-Hessenberg matrix and the HR algorithm on an $n \times n$ tridiagonal sign-symmetric matrix is discussed in [21].

It is not recommended to use the SR algorithm just by itself for solving a Hamiltonian eigenproblem, as it is potentially unstable. It is the method of choice in connection with the symplectic Lanczos method for Hamiltonian matrices, as will be discussed here. In case, one would like to use the SR algorithm, it should be accompanied by a defect-correction method like the Newton method in order to improve the accuracy of the computed results.

A few attempts have been made to create structure–preserving methods using a symplectic Lanczos method. The symplectic Lanczos method proposed by Mei [103] works with the squared Hamiltonian matrix and suffers from stability problems as well as from breakdown. There are several variants of symplectic Lanczos processes for Hamiltonian matrices available which create a Hamiltonian J-Hessenberg matrix, [17, 57, 139]. In [64], Freund and Mehrmann present a symplectic look–ahead Lanczos algorithm which overcomes breakdown by giving up the strict Hamiltonian J–Hessenberg form.

For any Hamiltonian matrix H the matrices H^2, $(H-\sigma I)^{-1}(H+\sigma I)^{-1}$ with $\sigma \in \mathbb{R}$ and $(H-\sigma I)^{-1}(H+\sigma I)^{-1}(H-\overline{\sigma} I)^{-1}(H+\overline{\sigma} I)^{-1}$ with $\sigma \in \mathbb{C}$ are skew-Hamiltonian matrices. The standard (implicitly restarted) Arnoldi method [126] automatically preserves this structure. This led to the development of the SHIRA method [7, 101] as a structure-preserving (shift-and-invert) Arnoldi method for Hamiltonian matrices.

In this paper we combine the ideas of implicitly restarted Lanczos methods [46, 66, 126] together with ideas to reflect the Hamiltonian structure and present a restarted symplectic Lanczos algorithm for the Hamiltonian eigenvalue problem. Implicitly restarted Lanczos methods typically have a higher numerical accuracy than explicit restarts and moreover they are more economical to implement [66].

A completely different class of algorithms is based on the matrix sign function, see, e.g., [12, 98, 123] and the references therein. Newton-like methods directed towards the computation or refinement of stable invariant subspaces for Hamiltonian matrices can be found in [68, 99, 98, 81].

Balancing can be a beneficial pre-processing step for computing eigenvalues of general matrices [105, 111]. Of course, standard balancing can be applied to a Hamiltonian matrix H as well; this, however, would destroy the structure of H and prevent the subsequent use of structure-preserving algorithms. Benner has developed in [13] a special-purpose balancing algorithm that is based on symplectic similarity transformations and thus preserves the structure of H. See also [22].

Structured condition numbers for invariant subspaces [45] and eigenvalues [74] of

structured matrices have been investigated as sometimes eigenvalues/invariant subspaces of structured matrices are better conditioned with respect to structured perturbations than with respect to general perturbations. The structured and the unstructured condition number for the stable invariant subspace of a Hamiltonian matrix are always the same [45], while there is no or little difference between the structured and the unstructured eigenvalue condition number [74].

This work is structured as follows. The notation and definitions used here are introduced in Section 2.1. The basic properties of Hamiltonian matrices are discussed. In Section 2.2 the general SR algorithm, the reduction of a general to J-Hessenberg form and the H algorithm are reviewed. The general nonsymmetric Lanczos algorithm is the subject of Chapter 2.3. Chapter 3 discussed the reduction of a Hamiltonian matrix to Hamiltonian J-Hessenberg form. In particular, the $4n-1$ parameters which determine a Hamiltonian J-Hessenberg from can be read off directly.

Chapter 4 discusses the SR algorithm for Hamiltonian matrices in detail. An equivalence between the HR and the Hamiltonian SR algorithm is given. As will be shown in Chapter 5, the SR algorithm for a Hamiltonian J-Hessenberg matrix H can be rewritten in a parameterized form that will work only with the $4n-1$ parameters which determine H instead of the entire matrix in each iteration step. Thus only $\mathcal{O}(n)$ flops per SR step are needed compared to $\mathcal{O}(n^3)$ flops when working on the actual Hamiltonian matrix. The key to the development of a SR algorithm working only on the parameters is the observation that at any point in the implicit SR step only a certain, limited number of rows and columns of the Hamiltonian J-Hessenberg matrix is worked on. In the leading part of the intermediate matrices the Hamiltonian J-Hessenberg form is already retained and is not changed any longer, while the trailing part has not been changed yet. Hence, from the leading part the first parameters of the resulting J-Hessenberg matrix can be read off, while from the trailing part the last parameters of the original J-Hessenberg matrix can still be read off. Our goal will be to derive the new parameters directly from the original ones without ever forming the actual Hamiltonian J-Hessenberg matrix or the bulge which is chased in the SR step.

Due to the special Hamiltonian eigenstructure, the spectral transformation function will be chosen either as

$$q_2(H) = (H - \mu I)(H + \mu I), \qquad \mu \in \mathbb{R} \text{ or } \mu = i\omega, \omega \in \mathbb{R},$$

or

$$q_4(H) = (H - \mu I)(H + \mu I)(H - \overline{\mu} I)(H + \overline{\mu} I), \qquad \mu \in \mathbb{C}, \mathrm{Re}(\mu) \neq 0.$$

In case an exceptional shift step is needed in the SR algorithm, one might want to use a single shift

$$q_1(H) = H - \mu I, \qquad \mu \in \mathbb{R}.$$

Each of the three cases is treated separately in the Sections 5.1 - 5.3. The SR iteration proceeds until the problem has completely decoupled into Hamiltonian J-Hessenberg subproblems of size 2×2 or 4×4. In a final step each of these subproblems has to be transform ed into a form from which the eigenvalues can be read off. In [38], this was considered for the case that the small subproblems have no purely imaginary eigenvalues. As this cannot be assumed in the context of the implicitly restarted symplectic Lanczos method, the solution of these subproblems in the presence of

purely imaginary eigenvalues is discussed in Chapter 6. The computation of the associated eigenvectors is the topic of Section 6.1. Numerical experiments are reported in Chapter 7.

Chapter 8 deals with the symplectic Lanczos method for Hamiltonian matrices. The structure–preserving Lanczos method generates a sequence of matrices

$$S^{2n,2k} = [v_1, v_2, \ldots, v_k, w_1, w_2, \ldots, w_k] \in \mathbb{R}^{2n\times 2k}$$

satisfying

$$HS^{2n,2k} = S^{2n,2k}\widetilde{H}^{2k,2k} + \zeta_{k+1}v_{k+1}e_{2k}^T \tag{1.5}$$

where $\widetilde{H}^{2k,2k}$ is a $2k \times 2k$ Hamiltonian J–Hessenberg matrix. Without some form of re–J–orthogonalization the symplectic Lanczos method is numerically unstable (see Section 8.6.1 and the discussion there). Thus, the symplectic Lanczos method suffers from the same numerical difficulties as any other Lanczos–like algorithm. One approach to deal with the numerical difficulties of Lanczos-like algorithms is to implicitly restart the symplectic Lanczos factorization. This was first introduced by Sorensen [126] in the context of nonsymmetric matrices and the Arnoldi process. Usually only a small subset of the eigenvalues is desired. As the eigenvalues of the Hamiltonian J-Hessenberg matrices $H^{2k,2k}$ are estimates for the eigenvalues of H, the length $2k$ symplectic Lanczos factorization (1.5) may suffice if the residual vector r_{k+1} is small. The idea of restarted Lanczos algorithms is to fix the number of steps in the Lanczos process at a prescribed value k which is dependent on the required number of approximate eigenvalues. The purpose of the implicit restart is to determine initial vectors such that the associated residual vectors are tiny. Given (1.5), an implicit Lanczos restart computes the Lanczos factorization

$$H\breve{S}^{2n,2k} = \breve{S}^{2n,2k}\breve{H}^{2k,2k} + \breve{r}_{k+1}e_{2k}^T$$

which corresponds to the starting vector

$$\breve{s}_1 = p(H)s_1$$

(where $p(H) \in \mathbb{R}^{2n\times 2n}$ is a polynomial) without having to explicitly restart the Lanczos process with the vector $\breve{s}_1$. This process is iterated until the residual vector r_{k+1} is tiny. J–orthogonality of the k Lanczos vectors is secured by re–J–orthogonalizing these vectors when necessary. This idea will be investigated in Section 8.4. As the iteration progresses, some of the Ritz values may converge to eigenvalues of H long before the entire set of wanted eigenvalues have. These converged Ritz values may be part of the wanted or unwanted portion of the spectrum. In either case it is desirable to deflate the converged Ritz values and corresponding Ritz vectors from the unconverged portion of the factorization. If the converged Ritz value is wanted then it is necessary to keep it in the subsequent factorizations; if it is unwanted then it must be removed from the current and the subsequent factorizations. A short comment on locking and purging techniques to accomplish this is given in Section 8.4.1. Most of the complications in the purging and deflating algorithms come from the need to preserve the structure of the decomposition, in particular, to preserve the J-Hessenberg form and the zero structure of the vector e_{2k}^T. In [130], Stewart shows how to relax the definition of an Arnoldi decomposition such that the purging and deflating problems can be solved in a natural and efficient way. Since the method is centered

about the Schur decomposition of the Hessenberg matrix, the method is called the Krylov-Schur method. In Section 8.5, a Krylov-Schur-like method for the symplectic Lanczos method is presented as first developed in [132].

But first, in Chapter 8, we will discuss the truncated symplectic Lanczos factorizations in Section 8.1, that is, first we are concerned with finding conditions for the symplectic Lanczos method terminating prematurely. This is a welcome event since in this case we have found an invariant symplectic subspace $S^{2n,2k}$ and the eigenvalues of $H^{2k,2k}$ are a subset of those of H. We will first discuss the conditions under which the residual vector of the symplectic Lanczos factorization will vanish at some step k. Then we will show how the residual vector and the starting vector are related. Finally a result indicating when a particular starting vector generates an exact truncated factorization is given. Stopping criteria which guarantee the required accuracy of the computed Ritz values and vectors are discussed in Section 8.2.

The symplectic Lanczos algorithm described above will, in general, compute approximations to a few of the largest eigenvalues of a Hamiltonian matrix H. Sometimes only a few of its smallest eigenvalues are needed. Since these are also the largest eigenvalues of H^{-1}, a Krylov subspace method can be applied to H^{-1} to find them. Since H^{-1} inherits the Hamiltonian structure of H, the symplectic Lanczos method is an appropriate method in the interest of efficiency, stability and accuracy. In situations where some prior information is given, one might prefer to use a shift before inverting. Specifically, if we know that the eigenvalues of interest lie near τ, we might prefer to work with $(H - \tau I)^{-1}$. Unfortunately, the shift destroys the Hamiltonian structure. Appropriate shift-and-invert strategies are discussed in Section 8.3. Numerical properties of the symplectic Lanczos algorithm are discussed in the final section of this chapter. Numerical experiments are given in Chapter 9.

2. Preliminaries.

2.1. Notations, Definitions, and Basic Properties. We will employ Householder notational convention. Capital and lower case letters denote matrices and vectors, respectively, while lower case Greek letters denote scalars. By $\mathbb{R}^{n\times k}$ we denote the real $n\times k$ matrices, by $\mathbb{C}^{n\times k}$ the complex $n\times k$ matrices. We use $\mathbb{K}$ to denote $\mathbb{R}$ or $\mathbb{C}$. The $n\times n$ identity matrix will be denoted by $I^{n,n}$, and the ith unit vector by e_i; $I^{n,n} = [e_1, e_2, \ldots, e_n]$. Let

$$J^{2n,2n} := \begin{bmatrix} 0 & I^{n,n} \\ -I^{n,n} & 0 \end{bmatrix} \tag{2.1}$$

and $P^{2n,2n}$ be the permutation matrix

$$P^{2n,2n} := [e_1, e_3, \ldots, e_{2n-1}, e_2, e_4, \ldots, e_{2n}] \in \mathbb{R}^{2n\times 2n}. \tag{2.2}$$

If the dimension of $I^{n,n}$, $J^{2n,2n}$, or $P^{2n,2n}$ is clear from the context, we leave off the superscript. We denote by $Z^{n,k}$ the first k columns of a $n\times n$ matrix Z.

We define an indefinite inner product by

$$(x, y)_J := x^T J y, \qquad x, y \in \mathbb{R}^{2n\times 2n}. \tag{2.3}$$

Let $A \in \mathbb{K}^{n\times k}$. Then we will denote

- the (i, j)th entry of A by a_{ij}
- the jth row of A by $A_{j,1:k}$
- the jth column of A by $A_{1:n,j}$

- the entries $\ell, \ell+1, \ldots, m$ of the jth row of A by $A_{j,\ell:m}$
- the entries $\ell, \ell+1, \ldots, m$ of the jth column of A by $A_{\ell:m,j}$
- the transpose of A by A^T; if $C = A^T$, then $c_{ij} = a_{ji}$

Corresponding notations will be used for vectors $x \in \mathbb{R}^n$.

Sometimes we partition the matrix $A \in \mathbb{K}^{n\times k}$ to obtain

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1q} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pq} \end{bmatrix} \begin{matrix} n_1 \\ \\ n_p \end{matrix} \qquad \begin{matrix} \\ \\ \\ \end{matrix} \tag{2.4}$$
$$\begin{matrix} k_1 & & k_q \end{matrix}$$

where $n_1 + \cdots + n_p = n, k_1 + \cdots + k_q = k$ and $A_{ij} \in \mathbb{K}^{n_i \times k_j}$ designates the (i,j) block or submatrix.

Throughout this thesis we will use the terms eigenvalues, spectrum, and invariant subspace as defined below.

DEFINITION 2.1. *Let $A \in \mathbb{K}^{n\times n}$.*

- *$\lambda \in \mathbb{C}$ is an eigenvalue of A if $\det(A - \lambda I) = 0$.*
- *$\sigma(A) = \{\lambda \in \mathbb{C} | \det(A - \lambda I) = 0\}$ is called the spectrum of A.*
- *$\mathcal{U} \subset \mathbb{C}^m$ determines an invariant subspace of A with respect to the eigenvalues in $\Lambda = \{\lambda_i | \lambda_i \in \mathbb{C}, i = 1, \ldots, k\}$ if there exist $U \in \mathbb{C}^{m\times k}$, and $K \in \mathbb{C}^{k\times k}$ such that U has full column rank, Λ is the spectrum of K, $AU = UK$ and the columns of U span $\mathcal{U}$. Sometimes we refer to U as the invariant subspace of A.*
- *A subspace $\mathcal{X} \subset \mathbb{R}^{2n}$ is called isotropic if $\mathcal{X} \perp J\mathcal{X}$. A maximal isotropic subspace is called Lagrangian.*
- *The spectral radius of A is defined by*

$$\rho(A) = \max\{|\lambda| : \lambda \in \sigma(A)\}.$$

We will make frequent use of the following notations.

- The Frobenius norm for $A \in \mathbb{K}^{n\times n}$ will be denoted by

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

- The 2–norm of a vector $x \in \mathbb{K}^n$ will be denoted by

$$||x||_2 = (x^T x)^{\frac{1}{2}}.$$

- The corresponding matrix norm for $A \in \mathbb{K}^{n\times n}$, the spectral norm, will be denoted by

$$||A||_2 = \rho(A^H A)^{\frac{1}{2}}.$$

- The condition number of a matrix $A \in \mathbb{K}^{n\times n}$ using the 2–norm is denoted by

$$\kappa_2(A) = ||A||_2 ||A^{-1}||_2.$$

- The rank of a matrix $A \in \mathbb{K}^{m\times n}$ will be denoted by

$$\text{rank}(A) = \dim(\text{ran}(A)),$$

where $\text{ran}(A)$ denotes the range of A

$$\text{ran}(A) = \{y \in \mathbb{K}^m : y = Ax \text{ for some } \ x \in \mathbb{K}^n\}.$$

- Given a collection of vectors $a_1, \ldots, a_n \in \mathbb{K}^m$, the set of all linear combinations of these vectors is a subspace referred to as the span of $\{a_1, \ldots, a_n\}$

$$\operatorname{span}\{a_1, \ldots, a_n\} = \left\{ \sum_{j=1}^{n} \beta_j a_j : \beta_j \in \mathbb{K} \right\}.$$

- The determinant of $A \in \mathbb{R}^{n \times n}$ is given by

$$\det(A) = \sum_{j=1}^{n} (-1)^{j+1} a_{1j} \det(A_{1j}).$$

  Here a_{1j} denote the entries of A in the first row and A_{1j} is an $(n-1) \times (n-1)$ matrix obtained by deleting the first row and jth column of A.
- The leading principal submatrix of order m of a matrix $A \in \mathbb{R}^{n \times n}$ is given by $A_{1:m,1:m}$. It will be denoted by $A^{m,m}$. The trailing principle submatrix of order m of a matrix $A \in \mathbb{R}^{n \times n}$ is given by $A_{(n-m+1):n,(n-m+1):n}$.
- The leading principal minor of order m of a matrix $A \in \mathbb{R}^{n \times n}$ is the determinant of the leading principal submatrix of order m, $\det(A_{1:m,1:m})$. The trailing principal minor of order m of a matrix $A \in \mathbb{R}^{n \times n}$ is the determinant of the trailing principal submatrix of order m, $\det(A_{(n-m+1):n,(n-m+1):n})$.

We use the following types of matrices and matrix factorization for matrices of size $n \times n$.

DEFINITION 2.2. *Let $A \in \mathbb{K}^{n \times n}, v \in \mathbb{K}^n$.*

- *A signature matrix is a diagonal matrix $D = \operatorname{diag}(d_1, \ldots, d_n)$ where $d_i \in \{\pm 1\}$.*
- *The matrix A is called D–symmetric if $(DA)^T = DA$ where D is a signature matrix.*
- *The Krylov matrix $K(A, v, j) \in \mathbb{K}^{n \times j}$ is defined by*

$$K(A, v, j) = [v, Av, \ldots, A^{j-1}v].$$

- *A is an upper Hessenberg matrix if $a_{ij} = 0$ for $i > j + 1, i, j = 1, \ldots, n$, that is,*

$$A = \begin{bmatrix} \ddots & \ddots \\ \ddots & \ddots \end{bmatrix}.$$

- *A is an unreduced upper Hessenberg matrix if A is an upper Hessenberg matrix with $a_{i,i-1} \neq 0, i = 2, \ldots, n$.*
- *A is an upper triangular matrix if $a_{ij} = 0$ for $i > j, i, j = 1, \ldots, n$, that is,*

$$A = \begin{bmatrix} \ddots & \ddots \\ & \ddots \end{bmatrix}.$$

- *A is a strict upper triangular matrix if A is an upper triangular matrix with $a_{ii} \neq 0, i = 1, \ldots, n$, that is,*

$$A = \begin{bmatrix} 0 & \ddots \\ \ddots & \ddots \\ & 0 \end{bmatrix}.$$

- *A is a quasi upper triangular matrix if it is a block matrix of the form (2.4) with blocks of size 1×1 or 2×2 and $A_{ij} = 0$ for $i > j, i = 1, \ldots, p, j = 1, \ldots, q$.*
- *A is a tridiagonal matrix if $a_{ij} = 0$ for $i > j+1$, and $i < j-1, i, j = 1, \ldots, n$, that is*

$$A = \begin{bmatrix} \ddots & \ddots & \\ \ddots & \ddots & \ddots \\ & \ddots & \ddots \end{bmatrix}.$$

- *A is an unreduced tridiagonal matrix if A is a tridiagonal matrix with $a_{i,i-1} \neq 0, i = 2, \ldots, n$ and $a_{i,i+1} \neq 0, i = 1, \ldots, n-1$.*
- *A is an orthogonal matrix, if $\mathbb{K} = \mathbb{R}$ and $A^T A = I$.*
- *A is a unitary matrix, if $\mathbb{K} = \mathbb{C}$ and $A^H A = I$.*
- *The QR factorization of $A \in \mathbb{K}^{n\times n}$ is given by $A = QR$ where $Q \in \mathbb{R}^{n\times n}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular if $\mathbb{K} = \mathbb{R}$. If $\mathbb{K} = \mathbb{C}$, then $Q \in \mathbb{C}^{n\times n}$ is unitary and $R \in \mathbb{C}^{n\times n}$ is upper triangular.*

LEMMA 2.3. *A tridiagonal matrix T is D–symmetric for some D if and only if $|t_{i+1,i}| = |t_{i,i+1}|$ for $i = 1, \ldots, n-1$. Every unreduced tridiagonal matrix is similar to a D–symmetric matrix (for some D) by a diagonal similarity with positive main diagonal entries.*

D–symmetric tridiagonal matrices are, e.g., generated by the nonsymmetric Lanczos process [82].

Hessenberg matrices play a fundamental role for the analysis of the standard eigenvalue algorithms considered in this thesis. Hence, let us review some of their most important properties. It is well-known that for any matrix $A \in \mathbb{R}^{n\times n}$ an orthogonal transformation matrix $Q \in \mathbb{R}^{n\times n}$ can be computed such that $Q^T AQ = H$ is of Hessenberg form (see, e.g., [65, Section 7.4.3]). Such a Hessenberg decomposition is not unique.

THEOREM 2.4 (Implicit-Q-Theorem). *Suppose $Q = [q_1, \ldots, q_n]$ and $V = [v_1, \ldots, v_n]$ are orthogonal matrices with the property that both $Q^T AQ = H$ and $V^T AV = G$ are upper Hessenberg matrices where $A \in \mathbb{R}^{n\times n}$. Let k denote the smallest positive integer for which $h_{k+1,k} = 0$, with the convention that $k = n$ if H is unreduced. If $q_1 = v_1$, then $q_i = \pm v_i$ and $|h_{i,i-1}| = |g_{i,i-1}|$ for $i = 2 : k$. Moreover, if $k < n$, then $g_{k+1,k} = 0$.*

Proof. See, e.g, [65, Theorem 7.4.2]. √

There is a useful connection between the Hessenberg reduction $Q^T AQ = H$ and the QR factorization of the Krylov matrix $K(A, Q(:, 1), n)$.

THEOREM 2.5. *Suppose $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix and $A \in \mathbb{R}^{n\times n}$. Let $q_1 = Qe_1$ be the first column of Q. Then $Q^T AQ = H$ is an unreduced upper Hessenberg matrix if and only if $Q^T K(A, q_1, n) = R$ is nonsingular and upper triangular.*

Proof. See, e.g, [65, Theorem 7.4.3]. √

Thus, there is a correspondence between nonsingular Krylov matrices and orthogonal similarity reductions to unreduced upper Hessenberg form.

The last two results mentioned here concern unreduced upper Hessenberg matrices. The left and right eigenvectors of unreduced upper Hessenberg matrices have the following properties.

THEOREM 2.6. *Suppose $H \in \mathbb{R}^{n\times n}$ is an unreduced upper Hessenberg matrix. If $Hs = \lambda s$ with $s \in \mathbb{K}^n \backslash \{0\}$ and $H^T u = \lambda u$ with $u \in \mathbb{K}^n \backslash \{0\}$, then $e_n^T s \neq 0$ and $e_1^T u \neq 0$.*

Proof. See, e.g, [86, Lemma 2.1]. √

Moreover, unreduced Hessenberg matrices are nonderogatory, that is, each eigenvalue has unit geometric multiplicity.

THEOREM 2.7. *Suppose $H \in \mathbb{R}^{n\times n}$ is an unreduced upper Hessenberg matrix. If λ is an eigenvalue of H, then its geometric multiplicity is one.*

Proof. See, e.g, [65, Theorem 7.4.4]. √

When there is a repeated eigenvalue, the theorem implies that H has less then n linearly independent eigenvectors. If the eigenvectors of a matrix of order n are not a basis for $\mathbb{R}^n$, then the matrix is called defective or nonsimple. Hence, if H has a repeated eigenvalue, it is a defective matrix. Unreduced Hessenberg matrices reveal even more information about the underlying eigensystem. Parlett [109, 110] provides an abundance of results for Hessenberg matrices.

For matrices of size $2n \times 2n$, we use the following types of matrices and matrix factorization.

DEFINITION 2.8. *Let $A \in \mathbb{R}^{2n\times 2n}$, $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$ where $A_{ij} \in \mathbb{R}^{n\times n}$ for $i, j = 1, 2$. Let $v \in \mathbb{R}^{2n}$.*

- *A is a J–Hessenberg matrix if A_{11}, A_{21}, A_{22} are upper triangular matrices and A_{12} is an upper Hessenberg matrix, that is*

$$A = \begin{bmatrix} \diagdown & \diagdown \\ \diagdown & \diagdown \end{bmatrix}.$$

- *A is an unreduced J–Hessenberg matrix if A is a J–Hessenberg matrix, A_{21}^{-1} exists, and A_{12} is an unreduced upper Hessenberg matrix.*
- *A is an (upper) J–triangular matrix if A_{11}, A_{12}, A_{22} are upper triangular matrices and A_{21} is a strict upper triangular matrix, that is,*

$$A = \begin{bmatrix} \diagdown & \diagdown \\ 0 \diagdown & \diagdown \\ \ddots & \\ 0 & \end{bmatrix}.$$

- *A is a lower J–triangular matrix if A^T is an upper J–triangular matrix.*
- *A is a symplectic matrix if $A^T J A = J$.*
- *A is a Hamiltonian matrix if $AJ = (AJ)^T$.*
- *A is a trivial matrix, if A is symplectic and J–triangular.*
- *The SR factorization of A is given by $A = SR$ where $S \in \mathbb{R}^{2n\times 2n}$ is symplectic and $R \in \mathbb{R}^{2n\times 2n}$ is J–triangular.*

Symplectic matrices can be viewed as orthogonal with respect to $(\cdot,\cdot)_J$. To emphasize this point of view, symplectic matrices are also called J–orthogonal.

The SR decomposition has been first introduced by Della-Dora [50, 51]. In contrast to the QR decomposition it does not always exist (see Theorem 2.10 below); but the set of matrices which can be factorized in this way is dense in $\mathbb{R}^{2n\times 2n}$ [36, 53]. While the QR decomposition is usually considered for matrices in $\mathbb{R}$ and $\mathbb{C}$, the SR decomposition is usually not considered for complex matrices $A \in \mathbb{C}^{2n\times 2n}$. This is due to the fact that the set of matrices $A \in \mathbb{C}^{2n\times 2n}$ which have an SR decomposition $A = SR$, where $S^H JS = J$ or $S^H JS = -J$, is not dense in $\mathbb{C}^{2n\times 2n}$ [36].

The following facts are easy to see.

LEMMA 2.9. *Let $A, B \in \mathbb{R}^{2n\times 2n}$, $A = \left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$ where $A_{ij} \in \mathbb{R}^{n\times n}$ for $i, j = 1, 2$. Let P be as in (2.2).*

a) If A is a J–triangular matrix, then PAP^T is an upper triangular matrix.

b) If A is a J–Hessenberg matrix or an unreduced J–Hessenberg matrix, then PAP^T is an upper Hessenberg matrix or an unreduced upper Hessenberg matrix, respectively.

c) A is trivial (that is, symplectic and J–triangular) if and only if it has the form

$$A = \begin{bmatrix} C & F \\ 0 & C^{-1} \end{bmatrix}, \tag{2.5}$$

where $C = \operatorname{diag}(c_1, \ldots, c_n)$, $F = \operatorname{diag}(f_1, \ldots, f_n)$.

d) If A is a regular J–triangular matrix, then A^{-1} is a J–triangular matrix.

e) If A and B are J–triangular matrices, then AB is a J–triangular matrix.

f) If A is a J–Hessenberg matrix and B a J–triangular matrix, then AB and BA are J–Hessenberg matrices.

Almost every matrix A can be decomposed into a product of a symplectic matrix S and a J–triangular matrix R.

THEOREM 2.10. *Let $A \in \mathbb{R}^{2n\times 2n}$ be nonsingular. There exists a symplectic matrix S and a J–triangular matrix R such that $A = SR$ if and only if all leading principal minors of even dimension of PA^TJAP^T are nonzero where P as in (2.2). The set of $2n \times 2n$ SR decomposable matrices is dense in $\mathbb{R}^{2n\times 2n}$.*

Proof. See [53, Theorem 11] or [36, Theorem 3.8] for a proof. √

Bunse-Gerstner and Mehrmann [38] present an algorithm for computing the SR decomposition of an arbitrary $2n \times 2n$ matrix A (see also Section 2.2.2). First-order componentwise and normwise perturbation bounds for the SR decomposition can be found in [47, 29].

The SR decomposition has properties similar to the QR decomposition.

COROLLARY 2.11. *Let $A \in \mathbb{R}^{2n\times 2n}$.*

a) If $A^T = SR$ is an SR decomposition of A^T, then there exists a symplectic matrix W and a lower J–triangular matrix L such that $A = LW$.

b) Let

$$\widehat{J} = \begin{bmatrix} & & & 1 \\ & & \iddots & \\ & 1 & & \\ 1 & & & \end{bmatrix}, \qquad \widetilde{J} = \begin{bmatrix} & & & \widehat{J} \\ & & \iddots & \\ & -\widehat{J} & & \\ -\widehat{J} & & & \end{bmatrix}.$$

If $\widetilde{J}A\widetilde{J} = SR$ is an SR decomposition of $\widetilde{J}A\widetilde{J}$, then there exists a symplectic matrix $\widetilde{S}$ and a lower J–triangular matrix $\widetilde{L}$ such that $A = \widetilde{S}\widetilde{L}$.

c) If $A^T = \widetilde{S}\widetilde{L}$ as in b) exists, then there exists a symplectic matrix $\widetilde{W}$ and an upper J–triangular matrix $\widetilde{R}$ such that $A = \widetilde{R}\widetilde{W}$.

Proof. See [55, Proposition 2.8] √

Let $X = \widetilde{J}A\widetilde{J}$. The SR decomposition of X exists if and only if the leading principal minors of even dimension of PX^TJXP^T are nonzero. We have

$$PX^TJXP^T = P\widetilde{J}A^TJA\widetilde{J}P^T$$

and

$$P\widetilde{J} = [-e_{2n}, -e_{2n-2}, \ldots, -e_2, e_{2n-1}, e_{2n-3}, \ldots, e_1].$$

Let $Y = A^T JA = \left[\begin{smallmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{smallmatrix}\right] \in \mathbb{R}^{2n\times 2n}$, where $Y_{ij} \in \mathbb{R}^{n\times n}$. Then the leading principal minors of even dimension of $PX^T JXP^T$ are given by

$$\det \begin{bmatrix} (Y_{11})_{2k+1:2n,2k+1:2n} & (Y_{12})_{2k+1:2n,2k+1:2n} \\ (Y_{21})_{2k+1:2n,2k+1:2n} & (Y_{22})_{2k+1:2n,2k+1:2n} \end{bmatrix}.$$

Hence, the leading principal minors of even dimension of $P(\widetilde{J}A\widetilde{J})^T J(\widetilde{J}A\widetilde{J})P^T$ are just the trailing principal minors of even dimension of $PA^T JAP^T$.

Statements similar to the above have been shown for the QR decomposition; see, e.g., [33, Korollar 2.5.2].

Symplectic matrices may serve to transform a $2n \times 2n$ matrix A to J–Hessenberg form. The relation between this transformation to J–Hessenberg form and the SR factorization is completely analogous to the relation between the unitary similarity reduction to Hessenberg form and the QR factorization, as the following theorem shows.

THEOREM 2.12 (Implicit-S-Theorem). *Let $A \in \mathbb{R}^{2n\times 2n}$.*

a) Let $A = SR$ and $A = \widetilde{S}\widetilde{R}$ be SR factorizations of A. Then there exists a trivial matrix D such that $\widetilde{S} = SD^{-1}$ and $\widetilde{R} = DR$.

b) Suppose $S \in \mathbb{R}^{2n\times 2n}$ is a symplectic matrix. Let $s_1 = Se_1$ be the first column of S. Then $S^{-1}AS$ is an unreduced J–Hessenberg matrix if and only if $S^{-1}K(A, s_1, 2n) = R$ is nonsingular and J–triangular.

Proof.

a) See, e.g., [38, Proposition 3.3].

b) See, e.g., [38, Theorem 3.4].

√

The essential uniqueness of the factorization $K(A, Se_1, n) = SR$ tells us that the transforming matrix S for the similarity transformation $S^{-1}AS$ is essentially uniquely determined by its first column. This Implicit-S-Theorem can serve as the basis for the construction of an implicit SR algorithm for J–Hessenberg matrices, just as the Implicit-Q-Theorem 2.4 provides a basis for the implicit QR algorithm on upper Hessenberg matrices. In both cases uniqueness depends on the unreduced character of the matrix. While the QR decomposition of a matrix always exists, the SR factorization may not exist. Hence, the reduction to J–Hessenberg form may not exist.

Unreduced J–Hessenberg matrices have properties similar to unreduced Hessenberg matrices.

THEOREM 2.13. *Suppose $H \in \mathbb{R}^{2n\times 2n}$ is an unreduced J–Hessenberg matrix.*

a) If $Hs = \lambda s$ with $s \in \mathbb{K}^{2n}\backslash\{0\}$ and $H^T u = \lambda u$ with $u \in \mathbb{K}^{2n}\backslash\{0\}$, then $e_n^T s \neq 0$ and $e_1^T u \neq 0$.

b) If λ is an eigenvalue of H, then its geometric multiplicity is one.

Proof. As $H_P = PHP^T$ with P as in (2.2) is an unreduced upper Hessenberg matrix, the theorem follows immediately from Theorem 2.6 and Theorem 2.7. √

REMARK 2.14. *If $Hx = \lambda x$, then $(Jx)^T H = -\lambda(Jx)^T$. Let y be the right eigenvector of H to $-\lambda$: $Hy = -\lambda y$, then $(Jy)^T H = \lambda(Jy)^T$. From Theorem 2.13 it follows*

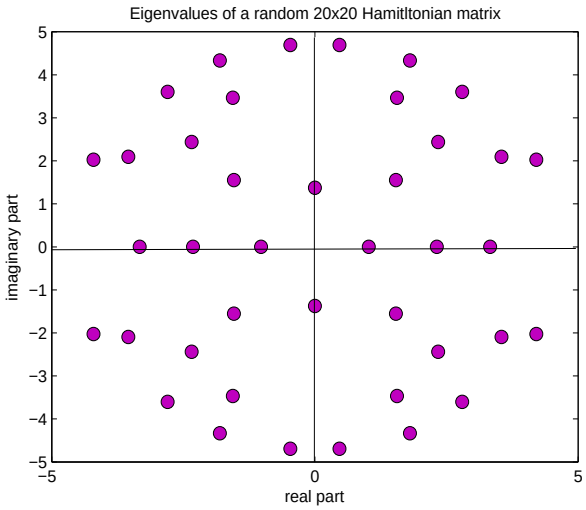


FIG. 2.1. *Eigenvalues of a Hamiltonian matrix*

that $e_{2n}^T y \neq 0$*, hence the* n*th component of the left eigenvector of* H *corresponding to* λ *is* $\neq 0$*.*

Recall that a matrix $H \in \mathbb{R}^{2n\times 2n}$ is called Hamiltonian if

$$(HJ)^T = HJ \tag{2.6}$$

while a matrix $M \in \mathbb{R}^{2n\times 2n}$ is called symplectic (or J–orthogonal) if

$$MJM^T = J \tag{2.7}$$

(or equivalently, $M^T JM = J$) where J is as in (2.1). In other words, the set $\mathcal{S}$ of all symplectic matrices is the set of all matrices that preserve the bilinear form defined by J. Symplectic matrices are nonsingular ($M^{-1} = JM^T J^T$). It is well-known and easy to show from this definition that $\mathcal{S}$ forms a multiplicative group (even more, $\mathcal{S}$ is a Lie group), while the set $\mathcal{H}$ of all Hamiltonian matrices of order $2n$ forms a Lie algebra. Moreover, since

$$(S^{-1}HS)J = S^{-1}HJS^{-T} = S^{-1}J^T H^T S^{-T} = [(S^{-1}HS)J]^T,$$

we see that symplectic similarity transformations preserve Hamiltonian structure. There are relevant cases, however, where both H and $S^{-1}HS$ are Hamiltonian but S is not a symplectic matrix [60].

The spectrum of a Hamiltonian matrix is symmetric with respect to the real and the imaginary axis (see Figure 2.1). If λ is an eigenvalue of H with right eigenvector x, then $-\lambda$ is an eigenvalue of H with left eigenvector $(Jx)^T$. Hence, as H is a real matrix, the eigenvalues always occur in pairs $\{\lambda, -\lambda\}$ if λ is real or purely imaginary, or in quadruples $\{\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}\}$ if λ is complex with nonzero real part.

In most applications, conditions are satisfied which guarantee the existence of an n–dimensional invariant subspace corresponding to the eigenvalues of H in the open left half plane. This is the subspace one usually wishes to compute. The numerical

computation of such an invariant subspace is typically carried out by an iterative procedure like the QR algorithm which transforms H into Schur form, from which the invariant subspace can be read off. See, e.g., [83, 84, 98]. For Hamiltonian matrices a special generalized Schur form is known.

THEOREM 2.15. *Let H be a $2n \times 2n$ real Hamiltonian matrix. Then there exists a real orthogonal and symplectic matrix Z such that*

$$Z^T H Z = \begin{bmatrix} T_1 & T_2 \\ 0 & -T_1^T \end{bmatrix}$$

where T_1 is an $n \times n$ real quasi upper triangular matrix, and T_2 is symmetric, if and only if every purely imaginary eigenvalue λ of H has even algebraic multiplicity, say $2k$, and any basis $X_k \in \mathbb{C}^{2n \times 2k}$ of the maximal invariant subspace for H corresponding to λ satisfies that $X_k^H J X_k$ is congruent to $J^{2k,2k}$. Moreover, Z can be chosen such that T_1 has only eigenvalues in the open left half plane.

Proof. This results was first stated and proved [91]. A simpler proof has been given in [92]. Weaker versions of the theorem assuming that H has no eigenvalues on the imaginary axis can be found, e.g., in [108]. A constructive proof of a weaker version can be found in [140], which was adapted from the construction given in [48]. √

The first n columns of the right transformation matrix Z then span an invariant subspace corresponding to the eigenvalues in the open left half plane. This subspace is unique if no eigenvalues are on the imaginary axis. The construction of numerical methods to compute these Schur forms using only symplectic and orthogonal transformations is still an open problem. See Section 1 for a summary of proposed methods.

A complete parameterization of all possible Hamiltonian Schur forms and corresponding Lagrange invariant subspaces has been given in [60]. For a discussion of other Hamiltonian canonical forms (e.g., symplectic Jordan or Kronecker canonical forms) see, e.g., [102, 85, 2].

2.2. The SR algorithm and the reduction to J-Hessenberg form. General QR like methods, in which the QR decompositions are replaced by other decompositions have been studied by several authors, see, e.g., [50, 142]. The factorizations have to satisfy several conditions to lead to a reasonable computational process. The one that meets most of these requirements for the Hamiltonian eigenproblem is the SR decomposition. This decomposition can serve as a basis for a QR like method, the SR algorithm, which works for arbitrary matrices of even dimensions. It preserves the Hamiltonian structure and, as will be seen, allows to develop fast and efficient implementations.

The SR algorithm [50, 51, 38] is a member of the family of GR algorithms [142] for calculating eigenvalues and invariant subspaces of matrices. The oldest member of the family is Rutishauser's LR algorithm [117, 118] and the most widely used is the QR algorithm [59, 77, 143, 137, 65, 138]. The GR algorithm is an iterative procedure that begins with a matrix A whose eigenvalues and invariant subspaces are sought. It produces a sequence of similar matrices (A_i) that (hopefully) converge to a form exposing the eigenvalues. The transforming matrices for the similarity transformations $A_i = G_i^{-1} A_{i-1} G_i$ are obtained from a "GR" decomposition $p_i(A_{i-1}) = G_i R_i$ in which p_i is a polynomial and R_i is upper triangular. The degree of p_i is called the multiplicity of the ith step. If p_i has degree 1, it is a single step. If the degree is 2, it is a double step,

and so on. Writing p_i in factored form $p_i(A) = \alpha_i(A-\mu_1^{(i)}I)(A-\mu_2^{(i)}I)\cdots(A-\mu_{m_i}^{(i)}I)$ we call the roots $\mu_1^{(i)}, \mu_2^{(i)}, \ldots, \mu_{m_i}^{(i)}$ the shifts for the ith step. Each step of multiplicity m_i has m_i shifts. A procedure for choosing the p_i is called a shift strategy because the choice of p_i implies a certain choice of shifts $\mu_1^{(i)}, \ldots, \mu_{m_i}^{(i)}$. In [142] it is shown that every GR algorithm is a form of a nested subspace iteration in which a change of coordinate system is made at each step. Convergence theorems for the GR algorithm are proved. The theorems guarantee convergence only if the condition numbers of the accumulated transforming matrices $\widehat{G}_i = G_1 G_2 \cdots G_i$ remain bounded throughout the iterations. The global convergence theorem holds for shift strategies that converge – unfortunately, no one has yet been able to devise a practical shift strategy that is guaranteed to converge for all matrices and can be shown to converge rapidly. The local convergence rate for the generalized Rayleigh-quotient strategy is typically quadratic. For matrices having certain types of special structure, it is cubic. In the generalized Rayleigh-quotient strategy p_i is chosen to be the characteristic polynomial of the trailing $m_i \times m_i$ submatrix of A_{i-1}.

Algorithms in the GR family are usually implemented implicitly, as chasing algorithms. The matrix whose eigenvalues are sought is first reduced to some upper Hessenberg-like form. Then the chasing algorithm is set in motion by a similarity transformation that introduces a bulge in the Hessenberg-like form near the upper left-hand corner of the matrix. A sequence of similarity transformations then chases the bulge downward and to the right, until the Hessenberg-like form is restored. Chasing steps like this are repeated until (hopefully) the matrix converges to a form from which the eigenvalues can be read off. A GR step consists of a similarity transformation $X = G^{-1}AG$ where $p(A) = GR$. One can show that G is more or less uniquely determined by its first column (e.g., in the QR step this follows from the Implicit-Q-Theorem). The implicit GR algorithm performs a different similarity transformation $\widetilde{X} = \widetilde{G}^{-1}A\widetilde{G}$, but $\widetilde{G}$ is constructed in such a way that its first column is proportional to the first column of G. It follows from the Implicit-G-Theorem that G and $\widetilde{G}$ are essentially the same, and consequently X and $\widetilde{X}$ are essentially the same. Watkins and Elsner analyze general GR chasing algorithms in [141].

2.2.1. Elementary Symplectic Matrices. During the course of the discussion of the SR algorithm we will use the following elementary symplectic transformations:

- Symplectic Givens transformations $G_k = G(k, c, s)$

$$G_k = \left[\begin{array}{cc|cc} I^{k-1,k-1} & & & \\ & c & & s \\ & & I^{n-k,n-k} & \\ \hline & & I^{k-1,k-1} & \\ & -s & & c \\ & & & I^{n-k,n-k} \end{array}\right],$$

  where $c^2 + s^2 = 1,\ c, s \in \mathbb{R}$.

- Symplectic Householder transformations $H_k = H(k, v)$

$$H_k = \left[\begin{array}{cc|cc} I^{k-1,k-1} & & & \\ & P & & \\ \hline & & I^{k-1,k-1} & \\ & & & P \end{array}\right],$$

  where $P = I^{n-k+1,n-k+1} - 2\frac{vv^T}{v^Tv},\ v \in \mathbb{R}^{n-k+1}$.

- Symplectic Gauss transformations $L_k = L(k, c, d)$

$$L_k = \left[\begin{array}{ccc|ccc} I^{k-2,k-2} & & & & & \\ & c & & & d & \\ & & c & & & \\ & & & I^{n-k,n-k} & & \\ \hline & & & I^{k-2,k-2} & & \\ & & & & c^{-1} & \\ & & & & & c^{-1} \\ & & & & & & I^{n-k,n-k} \end{array}\right],$$

  where $c, d \in \mathbb{R}$.
- Symplectic Gauss transformations type II $\widetilde{L}_k = \widetilde{L}(k, c, d)$

$$\widetilde{L}_k = \left[\begin{array}{ccc|ccc} I^{k-1,k-1} & & & & & \\ & c & & & d & \\ & & I^{n-k,n-k} & & & \\ \hline & & & I^{k-1,k-1} & & \\ & & & & c^{-1} & \\ & & & & & I^{n-k,n-k} \end{array}\right],$$

  where $c, d \in \mathbb{R}$.

The symplectic Givens and Householder transformations are orthogonal, while the symplectic Gauss transformations are nonorthogonal. It is crucial that the simple structure of these elementary symplectic transformations is exploited when computing matrix products of the form $G_k A, AG_k, H_k A, AH_k, L_k A, AL_k, \widetilde{L}_k A$, and $A\widetilde{L}_k$. Note that only rows k and $n+k$ are affected by the premultiplication $G_k A$, and columns k and $n+k$ by the postmultiplication AG_k. Similar, pre- and postmultiplication by L_k affects only the rows (resp., the columns) $k-1, k, n+k-1$ and $n+k$, while pre- and postmultiplication by $\widetilde{L}_k$ affects only the rows (resp., the columns) k and $n+k$. Premultiplication by H_k affects only the rows k to n and $n+k$ to $2n$, while postmultiplication affects the corresponding columns. Further, note that for the symplectic Householder transformations we have, e.g., $PA_{k:n,1:n} = A_{k:n,1:n} + vw^T$ where $w = \beta A_{k:n,1:n}^T v, \beta = -2/v^T v$. Thus a symplectic Householder update involves only matrix-vector multiplications followed by an outer product update. Failure to recognize these points and to treat the elementary symplectic transformations as general matrices increases work by an order of magnitude. The updates never entail the explicit formation of the transformation matrix, only the relevant parameters are computed. Algorithms to compute these parameters of the abovementioned transformations are given here for the sake of completeness (see Table 2.1 – 2.4 (in MATLAB[2]-like notation), see also, e.g., [108, 39]).

The Gauss transformations are computed such that among all possible transformations of that form, the one with the minimal condition number is chosen. The following lemma is easy to see.

LEMMA 2.16. *Let $M \in \mathbb{R}^{2n\times 2n}$ and $j, k \in \mathbb{N}$, $1 \le j \le 2n$, $1 \le k \le n$ given indices.*

*a) Let $[c, s] =$ **givens**$(M(k, j), M(k+n, j))$ and $G_k = G(k, c, s)$, then*

$$(G_k M)_{k+n,j} = 0.$$

[2]MATLAB is a trademark of The MathWorks, Inc.

Further G_k is symplectic and orthogonal.

b) Let $[c,s] = \mathbf{givens}(M(j,k+n), M(j,k))$ and $G_k = G_k(k,c,s)$, then

$$(MG_k)_{j,k} = 0.$$

Further G_k is symplectic and orthogonal.

c) Let $v = \mathbf{house}(M(k:n,j))$ and $H_k = H(k,v)$, then

$$(H_k M)_{k+1:n,j} = 0.$$

Further H_k is symplectic and orthogonal.

d) Let $v = \mathbf{house}(M(j,k+n:2n))$ and $H_k = H(k,v)$, then

$$(MH_k)_{j,k+1+n:2n} = 0.$$

Further H_k is symplectic and orthogonal.

e) Let $k > 1$ and $M(k-1+n,j) = 0$ only if $M(k,j) = 0$. Let $L_k = L(k,c,d)$ where $[c,d,\kappa] = \mathbf{gauss1}(M(k,j), M(k-1+n,j))$, then

$$(L_k M)_{k,j} = 0.$$

Further L_k is symplectic with the condition number κ. κ is minimal, that is, there is no corresponding elimination matrix with smaller condition number.

f) Let $k > 1$ and $M(j,k-1) = 0$ only if $M(j,k+n) = 0$, $[c,d,\kappa] = \mathbf{gauss1}(M(j, k+n), M(j,k-1))$ and $L_k = L(k,c^{-1},d)$, then

$$(ML_k)_{j,k+n} = 0.$$

Further L_k is symplectic with condition number κ. κ is minimal, that is, there is no corresponding symplectic elimination matrix with smaller condition number.

g) Let $M(j,k) = 0$ only if $M(j,k+n) = 0$, $[c,d,\kappa] = \mathbf{gauss2}(M(j,k+n), M(j, k))$ and $\widetilde{L}_k = \widetilde{L}(k,c^{-1},d)$, then

$$(M\widetilde{L}_k)_{j,k+n} = 0.$$

Further $\widetilde{L}_k$ is symplectic with condition number κ. κ is minimal, that is, there is no corresponding symplectic elimination matrix with smaller condition number.

REMARK 2.17. *Any orthogonal symplectic matrix Q can be expressed as the product of symplectic Givens and symplectic Householder transformations, see [108, Corollary 2.2].*

REMARK 2.18. *In some of the algorithms discussed later on, the symplectic Householder transformation can be replaced by a symplectic Givens transformation of type II $\widetilde{G}_k = \widetilde{G}(k,c,s)$ which is defined as*

$$\widetilde{G}_k = \left[\begin{array}{cccc|cccc} I_{k-1} & & & & & & & \\ & c & s & & & & & \\ & -s & c & & & & & \\ & & & I_{n-k-2} & & & & \\ \hline & & & & I_{k-1} & & & \\ & & & & & c & s & \\ & & & & & -s & c & \\ & & & & & & & I_{n-k-2} \end{array}\right],$$

Algorithm: Generate Symplectic Givens Matrix

Given scalars a and b compute c and s such that $c^2 + s^2 = 1$ and

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

function $[c, s] =$ **givens**(a, b)
  if $b = 0$
  then $c = 1, s = 0$
  else if $|b| > |a|$
    then $t = a/b$
      $s = -1/\sqrt{1 + t^2}$
      $c = st$
    else $t = b/a$
      $c = 1/\sqrt{1 + t^2}$
      $s = ct$
    end
  end

TABLE 2.1
Symplectic Givens Matrix

Algorithm: Generate Symplectic Householder Matrix

Given a column vector $x \in \mathbb{R}^n$ compute v such that $v(1) = 1$,
and $y(2 : n) = 0$ for $y = (I - 2vv^T/(v^T v))x$.

function $v =$ **house**(x)
  $m = \|x\|_2$
  $v = x$
  if $m \neq 0$
  then if $x(1) \geq 0$
    then $b = x(1) + m$
    else $b = x(1) - m$
    end
    $v = (1/b)v$
  end
  $v(1) = 1$

TABLE 2.2
Symplectic Householder Matrix

where $c^2+s^2 = 1, c, s \in \mathbb{R}$. *Usually,* c *and* s *are computed as in the standard symplectic Givens transformation.*

2.2.2. SR **Algorithm.** The SR algorithm is based on the SR decomposition. Recall that the SR factorization of a real $2n \times 2n$ matrix A is given by $A = SR$ where $S \in \mathbb{R}^{2n\times 2n}$ is symplectic and $R \in \mathbb{R}^{2n\times 2n}$ is J–triangular. Almost every matrix A can be decomposed into such a product, see Theorem 2.10.

Algorithm: Generate Symplectic Gauss Matrix

Given scalars a and b, where $b = 0$ only if $a = 0$, compute c and d such that

$$\begin{bmatrix} c & & & d \\ & c & d & \\ & & c^{-1} & \\ & & & c^{-1} \end{bmatrix} \begin{bmatrix} \star \\ a \\ b \\ 0 \end{bmatrix} = \begin{bmatrix} \star \\ 0 \\ r \\ 0 \end{bmatrix}.$$

Further the condition number κ of the elimination matrix is computed.

function $[c, d, \kappa] =$ **gauss1**(a, b)
  if $a = 0$
  then $t = 0$
  else $t = -a/b$
  end
  $c = 1/\sqrt[4]{1 + t^2}$
  $d = ct$
  $\kappa = \sqrt{1 + t^2} + |t|$

TABLE 2.3
Symplectic Gauss Matrix

Algorithm: Generate Symplectic Gauss Matrix Type II

Given scalars a and b, where $b = 0$ only if $a = 0$, compute c and d such that

$$\begin{bmatrix} c & d \\ & c^{-1} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ r \end{bmatrix}.$$

Further the condition number κ of the elimination matrix is computed.

function $[c, d, \kappa] =$ **gauss2**(a, b)
  if $a = 0$
  then $t = 0$
  else $t = -a/b$
  end
  $c = 1/\sqrt[4]{1 + t^2}$
  $d = ct$
  $\kappa = \sqrt{1 + t^2} + |t|$

TABLE 2.4
Symplectic Gauss Matrix Type II

If the SR decomposition exists, then other SR decompositions of A can be built from it by passing trivial factors (2.5) back and forth between S and R. That is, if D is a trivial matrix, $\widetilde{S} = SD^{-1}$ and $\widetilde{R} = DR$, then $A = \widetilde{S}\widetilde{R}$ is another SR decomposition of A (see Theorem 2.12). If A is nonsingular, then this is the only way to create other SR decompositions. In other words, the SR decomposition is unique up to trivial factors.

In Section 2.1 we have already seen that the relation between the symplectic transformation of a $2n \times 2n$ matrix to J–Hessenberg form and the SR decomposition is completely analogous to the relation between the unitary reduction to upper Hessenberg form and the QR decomposition, see Theorem 2.12. The SR decomposition $A = SR$ and, therefore, also the reduction to J–Hessenberg form can, in general, not be performed with a symplectic orthogonal matrix S. A necessary and sufficient condition for the existence of such an orthogonal SR decomposition is that A is of the form

$$A = \begin{bmatrix} X & Y \\ -Y & X \end{bmatrix} R$$

where $X, Y \in \mathbb{R}^{n \times n}$, and R is a J–triangular matrix [36]. Hence, for the computation of the SR decomposition (or the reduction to J–Hessenberg form) one has to employ nonorthogonal symplectic transformations.

Bunse-Gerstner and Mehrmann [38] present an algorithm for computing the SR decomposition of an arbitrary $2n \times 2n$ matrix A. The algorithm uses the symplectic Givens transformations G_k, the symplectic Householder transformations H_k, and the symplectic Gauss transformation L_k introduced in Section 2.2.1. Symplectic elimination matrices S_j are determined such that $R = S_{2n} \cdots S_2 S_1 A$ is of J–triangular form. Then $A = SR$ with $S = S_1^{-1} S_2^{-1} \cdots S_{2n}^{-1}$ is an SR decomposition of A. The basic idea of the algorithm can be summarized as follows:

let $S = I$, $R = A$
for $j = 1$ to n
  determine a symplectic matrix S_{2j-1} such that the jth column
    of $S_{2j-1}R$ is of the desired form
  set $S = SS_{2j-1}^{-1}$, $R = S_{2j-1}R$
  determine a symplectic matrix S_{2j} such that the $(n+j)$th column
    of $S_{2j}R$ is of the desired form
  set $S = SS_{2j}^{-1}$, $R = S_{2j}R$

The entries $n+i$ to $2n$ of the ith column and the entries $n+i+1$ to $2n$ of the $(n+i)$th column are eliminated using symplectic Givens matrices. The entries $i+1$ to n of the ith column and the entries $i+2$ to n of the $(n+i)$th column are eliminated using symplectic Householder matrices. The entry $(n+i+1)$ of the $(n+i)$th column is eliminated using a symplectic Gauss matrix. This algorithm for computing the SR decomposition of an arbitrary matrix (as given in [38]) can be summarized as given in Table 2.5 (in MATLAB.

If at any stage $j \in \{1, \ldots, n-1\}$ the algorithms ends because of the stopping condition, then the $2j$th leading principal minor of PA^TJAP^T is zero and A has no SR decomposition (see Theorem 2.10).

All but $(n-1)$ transformations are orthogonal, which are known to be numerically stable transformations. Applying symplectic Gauss transformations for elimination, problems can arise not only because the algorithm may break down but also in those cases where we are near to such a breakdown. If we eliminate the jth nonzero entry of a vector x with a symplectic Gauss matrix L_j and x_{n-j-1} is very small relative to x_j, then the condition number $\kappa_2(L_j)$, here essentially given by

$$||L_j||_2 = (1 + v^2)^{1/2} + |v|,$$

where $v = -x_j/x_{n-j-1}$, will be very large. A transformation with L_j will then cause dramatic growth of the rounding errors in the result. Here we will always choose

Algorithm: SR Decomposition

Given a $2n \times 2n$ matrix A compute a $2n \times 2n$ symplectic matrix S, and a $2n \times 2n$ J–triangular matrix R such that $A = SR$.

$S = I^{2n,2n}$
for $j = 1 : n$
  for $k = n : -1 : j$
    compute G_k such that $(G_k A)_{k+n,j} = 0$
    $A = G_k A$
    $S = S G_k^T$
  end
  if $j < n$
  then compute H_j such that $(H_j A)_{j+1:n,j} = 0$.
    $A = H_j A$
    $S = S H_j^T$
    for $k = n : -1 : j+1$
      compute G_k such that $(G_k A)_{k+n,j+n} = 0$.
      $A = G_k A$
      $S = S G_k^T$
    end
    if $j < n-1$
    then compute H_{j+1} such that $(H_{j+1} A)_{j+2:n,j+n} = 0$.
      $A = H_{j+1} A$
      $S = S H_{j+1}^T$
    end
    if $A_{j+n,j+n} = 0$ and $A_{j+1,j+n} \neq 0$
    then stop, SR decomposition does not exist
    end
    compute L_{j+1} such that $(L_{j+1} A)_{j+1,j+n} = 0$.
    $A = L_{j+1} A$
    $S = S L_{j+1}^{-1}$
  end
end

TABLE 2.5
SR Decomposition

the symplectic Gauss matrix among all possible ones with optimal (smallest possible) condition number.

Using the SR decomposition the SR algorithm for an arbitrary $2n \times 2n$ matrix A is given as

let $A_0 = A$
for $k = 1, 2, \ldots$
  choose a shift polynomial p_k
  compute the SR decomposition $p_k(A_{k-1}) = S_k R_k$
  compute $A_k = S_k^{-1} A_{k-1} S_k$

The SR decomposition of $p_k(A_{k-1})$ might not exist. As the set of the matrices, for which the SR decomposition does not exist, is of measure zero (Theorem 2.10), the

polynomial p_k is discarded and an implicit SR step with a random shift is performed as proposed in [38]. For an actual implementation this might be realized by checking the condition number of the Gauss transformation L_k needed in each step and performing an exceptional step if it exceeds a given tolerance.

REMARK 2.19. *How does a small perturbation of A influence the SR step? Will the SR step on $A+E$, where E is an error matrix with small norm, yield a transformed matrix $S(A+E)S^{-1}$ close to SAS^{-1}? How does finite-precision arithmetic influence the SR step? As in the SR algorithm nonorthogonal symplectic similarity transformations are employed, a backward error analysis would yield*

$$S(A+E)S^{-1} = SAS^{-1} + G$$

where $||G||_2 = ||SES^{-1}||_2 \leq \kappa_2(S)||E||_2$. The condition number $\kappa_2(S)$ can be arbitrarily large. The QR algorithm does not have this problem. As in the QR algorithm only unitary similarity transformation are employed, an error analysis yields

$$Q(A+E)Q^T = QAQ^T + F$$

where $||F||_2 = ||QEQ^T||_2 = ||E||_2$. First-order componentwise and normwise perturbation bounds for the SR decomposition can be found in [47, 29].

The shift polynomials p_k are usually chosen according to the generalized Rayleigh-quotient strategy modified for the situation given here, that is, p_k is chosen to be the characteristic polynomial of the trailing $m_i \times m_i$ submatrix of $PA_{i-1}P^T$ where P as in (2.2). A convergence proof can be deduced from the corresponding proof of convergence for general GR algorithms in [142].

THEOREM 2.20. *Let $A_0 \in \mathbb{R}^{2n\times 2n}$, and let p be a polynomial. Let $\lambda_1, \ldots, \lambda_{2n}$ denote the eigenvalues of A_0, ordered so that $|p(\lambda_1)| \geq |p(\lambda_2)| \geq \ldots \geq |p(\lambda_{2n})|$. Suppose k is a positive integer less than $2n$ such that $|p(\lambda_k)| \geq |p(\lambda_{k+1})|$, let $\rho = |p(\lambda_{k+1})|/|p(\lambda_k)|$, and let (p_i) be a sequence of polynomials such that $p_i \to p$ and $p_i(\lambda_j) \neq 0$ for $j = 1, \ldots, k$ and all i. Let $\mathcal{U}$ be the invariant subspace of PA_0P^T associated with $\lambda_{k+1}, \ldots, \lambda_{2n}$, and suppose* $\operatorname{span}\{e_1, \ldots, e_k\} \cap \mathcal{U} = \{0\}$ *(P as in (2.2)). Let (A_i) be the sequence of iterates of the SR algorithm using these p_i, starting from A_0. If there exists a constant $\widehat{\kappa}$ such that the cumulative transformation matrices $\widehat{S}_i = S_1 S_2 \cdots S_i$ all satisfy $\kappa_2(\widehat{S}_i) \leq \widehat{\kappa}$, then (PA_iP^T) tends to block upper triangular form, in the following sense. Write*

$$PA_iP^T = \begin{bmatrix} X_{11}^{(i)} & X_{12}^{(i)} \\ X_{21}^{(i)} & X_{22}^{(i)} \end{bmatrix},$$

where $X_{11}^{(i)} \in \mathbb{R}^{k\times k}$. Then for every $\widehat{\rho}$ satisfying $\rho < \widehat{\rho} < 1$ there exists a constant C such that $||X_{21}^{(i)}||_2 \leq C\widehat{\rho}^{\,i}$ for all i.

Proof. See Theorem 6.2 in [142]. √

The condition $p_i(\lambda_j) \neq 0$ for $j = 1, \ldots, k$ may occasionally be violated. If $p_i(\lambda_j) = 0$, then $p_i(A_i)$ is singular. It can be shown that in this case, the eigenvalue λ_j can be deflated from the problem after the ith iteration. The theorem further implies that the eigenvalues of $X_{11}^{(i)}$ and $X_{22}^{(i)}$ converge to $\lambda_1, \ldots, \lambda_k$ and $\lambda_{k+1}, \ldots, \lambda_{2n}$, respectively.

REMARK 2.21. *The condition* $\operatorname{span}\{e_1, \ldots, e_k\} \cap \mathcal{U} = \{0\}$ *is automatically satisfied for all unreduced J–Hessenberg matrices. Suppose $x \in \operatorname{span}\{e_1, \ldots, e_k\}$ is*

nonzero. Let its last nonzero component be $x_r, r \leq k$. *If* A_0 *has unreduced* J–*Hessenberg form, then* PA_0P^T *is an unreduced upper Hessenberg matrix. The last nonzero component of* PA_0P^Tx *is its* $(r+1)$*st, the last nonzero component of* $PA_0^2P^Tx$ *is its* $(r+2)$*nd, and so on. It follows that* $x, A_0x, A_0^2x, \ldots, A_0^mx$ *are linearly independent, where* $m = 2n-k$. *Therefore the smallest invariant subspace of* PA_0P^T *that contains* x *has dimension at least* $m+1$. *Since* $\mathcal{U}$ *is invariant under* PA_0P^T *and has dimension* m, *it follows that* $x \notin \mathcal{U}$. *Thus* $\operatorname{span}\{e_1, \ldots, e_k\} \cap \mathcal{U} = \{0\}$.

The following theorem indicates quadratic and cubic converge under certain circumstances.

THEOREM 2.22. *Let* $A_0 \in \mathbb{R}^{2n\times 2n}$ *have distinct eigenvalues. Let* (A_i) *be the sequence generated by the* SR *algorithm starting from* A_0, *using the generalized Rayleigh-quotient shift strategy with polynomials of degree* $2m$. *Suppose there is a constant* $\widehat{\kappa}$ *such that* $\kappa_2(\widehat{S}_i) \leq \widehat{\kappa}$ *for all* i, *and the* PA_iP^T *converge to block triangular form, in the sense described in Theorem 2.20, with* $k = 2n-2m$. *Then the convergence is quadratic. Moreover, suppose that each of the iterates*

$$PA_iP^T = \begin{bmatrix} X_{11}^{(i)} & X_{12}^{(i)} \\ X_{21}^{(i)} & X_{22}^{(i)} \end{bmatrix}$$

satisfies $||X_{12}^{(i)}|| = ||X_{21}^{(i)}||$ *for some fixed norm* $||\cdot||$. *Then the iterates converge cubically if they converge.*

Proof. See Theorems 6.3 and 6.5 in [142]. √

Hamiltonian matrices satisfy the condition $||X_{12}^{(i)}|| = ||X_{21}^{(i)}||$, thus the convergence rate of the Hamiltonian SR algorithm is typically cubic [142, Example 6.7].

The most glaring shortcoming associated with the above algorithm is that each step requires a full SR decomposition costing $\mathcal{O}((2n)^3)$ flops. Fortunately, the amount of work per iteration can be reduced by an order of magnitude if we first reduce the full matrix A to J–Hessenberg form as the SR algorithm preserves the J–Hessenberg form: If $p_k(A_{k-1})$ is nonsingular and $p_k(A_{k-1}) = S_kR_k$, then R_k is nonsingular as S_k is symplectic. Therefore,

$$A_k = S_k^{-1}A_{k-1}S_k = R_kp_k(A_{k-1})^{-1}A_{k-1}p_k(A_{k-1})R_k^{-1} = R_kA_{k-1}R_k^{-1}$$

because $p_k(A_{k-1})$ and A_{k-1} commute. If A_{k-1} is of J–Hessenberg form, then so is A_k as A_k is a product of a J–Hessenberg matrix (A_{k-1}) and J-triangular matrices (R_k, R_k^{-1}). For singular $p_k(A_{k-1})$ one has to check the special form of S_k to see that A_k is of desired form if A_{k-1} is of J–Hessenberg form. In this case the problem can be split into two problems of smaller dimensions: If $\operatorname{rank}(p_k(A_{k-1})) = 2n-2\nu = 2j$, then the problem splits into a problem of size $2j \times 2j$ with J–Hessenberg form and a problem of size $2\nu \times 2\nu$ whose eigenvalues are exactly the shifts that are eigenvalues of A_{k-1} (that is, that are eigenvalues of A), see, e.g., [141, Section 4]. The SR decomposition of a J–Hessenberg matrix requires only $\mathcal{O}((2n)^2)$ flops to calculate as compared to $\mathcal{O}((2n)^3)$ flops for the SR decomposition of a full $2n\times 2n$ matrix. Hence, as the initial reduction to J–Hessenberg form is an $\mathcal{O}((2n)^3)$ process, a reasonable implementation of the SR algorithm should first reduce A to J–Hessenberg form.

Because of the essential uniqueness of the reduction to J–Hessenberg form, the SR algorithm can be performed without explicitly computing the decompositions $p_k(A_{k-1}) = S_kR_k$. In complete analogy to the GR algorithm, we can perform the SR step implicitly:

compute a symplectic matrix $\widetilde{S}_k$ such that $\widetilde{S}_k^{-1} p_k(A_{k-1}) e_1 = \alpha e_1$
for some $\alpha \in \mathbb{R}$
set $\widehat{A}_k = \widetilde{S}_k^{-1} A_{k-1} \widetilde{S}_k$
compute a symplectic matrix $\widehat{S}_k$ such that $\widehat{S}_k^{-1} \widehat{A}_k \widehat{S}_k$ is of
J–Hessenberg form

The resulting J–Hessenberg matrix $\widehat{S}_k^{-1} \widehat{A}_k \widehat{S}_k$ is essentially the same as $S_k^{-1} A_{k-1} S_k$, since $\widehat{S}_k = D S_k$ for some trivial matrix D (2.5).

Applying the first transformation $\widetilde{S}_k$ to the J–Hessenberg matrix A_{k-1} yields a matrix with almost J–Hessenberg form having a small bulge, that is there will be some additional entries in the upper left hand corner of each $n \times n$ block of $\widetilde{S}_k^{-1} A_{k-1} \widetilde{S}_k$. The remaining implicit transformations (that is, the computation of $\widehat{S}_k$) perform a bulge chasing sweep down the diagonal to restore the J–Hessenberg form.

Bunse-Gerstner and Mehrmann present in [38] an algorithm for reducing an arbitrary matrix to J–Hessenberg form. Depending on the size of the bulge in $\widetilde{S}_k^{-1} A_{k-1} \widetilde{S}_k$, the algorithm can be greatly simplified to reduce $\widetilde{S}_k^{-1} A_{k-1} \widetilde{S}_k$ to J–Hessenberg form. The algorithm uses the symplectic Givens transformations G_k, the symplectic Householder transformations H_k, and the symplectic Gauss transformations L_k introduced in Section 2.2.1. The basic idea of the algorithm can be summarized as follows:

for $j = 1$ to n
determine a symplectic matrix S such that the jth column of
$S^{-1}A$ is of the desired form
set $A = S^{-1} A S$
determine a symplectic matrix S such that the $(n+j)$th column
of $S^{-1}A$ is of the desired form
set $A = S^{-1} A S$

In order to compute a symplectic matrix S such that the jth column of $S^{-1}A$ is of the desired form the following actions are taken. The entries $n+j$ to $2n$ of the jth column are eliminated using symplectic Givens matrices. The entries $j+2$ to n of the jth column are eliminated using symplectic Householder matrices. The entry $(j+1)$ of the jth column is eliminated using a symplectic Gauss matrix. Similar, in order to compute a symplectic matrix S such that the $(n+j)$th column of $S^{-1}A$ is of the desired form the following actions are taken. The entries $n+j$ to $2n$ of the $(n+j)$th column are eliminated using symplectic Givens matrices. The entries $j+2$ to n of the $(n+j)$th column are eliminated using symplectic Householder matrices. This algorithm for computing the reduction of an arbitrary matrix to J–Hessenberg form (as given in [38]) can be summarized as given in Table 2.6 (in MATLAB-like notation). As the reduction is computed columnwise, starting with the first column, the first column of the matrix S will be a multiple of the first unit vector, that is $Se_1 = \alpha e_1$.

For some later discussions we will need a reduction to J-Hessenberg such that the last row of the reduction matrix S is a multiple of the $2n$th unit vector, that is $e_{2n}^T S = \alpha e_{2n}^T$. This has been discussed in [132]. The idea can be achieved when the matrix A is reduced rowwise beginning with the last row

for $j = 1$ to n
determine a symplectic matrix S such that the $(2n-j+1)$th
row of AS is of the desired form
set $A = S^{-1} A S$
determine a symplectic matrix S such that the jth row
of AS is of the desired form

Algorithm: Reduction to J–Hessenberg Form

Given a $2n \times 2n$ arbitrary matrix A compute its reduction to J–Hessenberg form. A will be overwritten by its J–Hessenberg form.

for $j = 1 : n-1$
  for $k = n : -1 : j+1$
    compute G_k such that $(G_k A)_{k+n,j} = 0$
    $A = G_k A G_k^T$
  end
  if $j < n-1$
  then compute H_j such that $(H_j A)_{j+2:n,j} = 0$
    $A = H_j A H_j^T$
  end
  if $A_{j+1,j} \neq 0$ and $A_{n+j,n+j} = 0$
  then stop, reduction does not exist
  end
  compute L_{j+1} such that $(L_{j+1} A)_{j+1,j} = 0$
  $A = L_{j+1} A L_{j+1}^{-1}$
  for $k = n : -1 : j+1$
    compute G_k such that $(G_k A)_{n+k,n+j} = 0$
    $A = G_k A G_k^T$
  end
  if $j < n-1$
  then compute H_j such that $(H_j A)_{j+2:n,n+j} = 0$
    $A = H_j A H_j^T$
  end
end

TABLE 2.6
Reduction to J–Hessenberg Form

set $A = S^{-1} A S$

In order to compute a symplectic matrix S such that the $(2n-j+1)$th row of AS is of the desired form the following actions are taken. The entries 1 to $n-j$ of the $(2n-j+1)$th row are eliminated using symplectic Givens matrices. The entries $n+1$ to $2n-j+1$ of the $(2n-j+1)$th row are eliminated using symplectic Householder matrices. The entry $(2n-j+1, 2n-j)$ of the $(2n-j+1)$th row is eliminated using a symplectic Gauss matrix. Similar, in order to compute a symplectic matrix S such that the $(n-j+1)$th row of AS is of the desired form the following actions are taken. The entries 1 to $n-j$ of the $(n-j+1)$th row are eliminated using symplectic Givens matrices. The entries $n+1$ to $2n-j-1$ of the $(n-j+1)$th row are eliminated using symplectic Householder matrices. This algorithm for computing the reduction of an arbitrary matrix to J–Hessenberg form (as given in [132]) can be summarized as given in Table 2.7 (in MATLAB-like notation). As the reduction is computed rowwise, starting with the last row, we have $e_{2n}^T S = \alpha e_{2n}^T$. Please note, that, e.g., the symplectic Householder matrix H defined before is constructed such that it transforms a given vector x to a multiple of the first unit vector. In the rowwise reduction as described here, we will need a symplectic Householder transformation which transforms a given

row vector y^T to a multiple of e_{2n}^T. Likewise, the symplectic Gauss elimination has to be defined in a slightly different way. In an abuse of notation, we are using the same notation H and L in the description of the rowwise algorithm as in the columnwise algorithm!

Algorithm: Rowwise reduction to J–Hessenberg Form

Given a $2n \times 2n$ arbitrary matrix A compute its reduction to J–Hessenberg form. A will be overwritten by its J–Hessenberg form.

for $j = 1 : n-1$
  for $k = 1 : n-j$
    compute G_k such that $(AG_k)_{2n-j+1,k} = 0$
    $A = G_k^T A G_k$
  end
  if $j < n-1$
  then compute H_j such that $(AH_j)_{2n-j+1,n+1:2n-j+1} = 0$
    $A = H_j^T A H_j$
  end
  if $A_{2n-j+1,2n-j} \neq 0$ and $A_{2n-j+1,n-j+1} = 0$
  then stop, reduction does not exist
  end
  compute L_{j+1} such that $(AL_{j+1})_{2n-j+1,2n-j} = 0$
  $A = L_{j+1}^{-1} A L_{j+1}$
  for $k = 1 : n-j$
    compute G_k such that $(AG_k)_{n-j+1,k} = 0$
    $A = G_k^T A G_k$
  end
  if $j < n-1$
  then compute H_j such that $(AH_j)_{n-j+1,n+1:2n-j-1} = 0$
    $A = H_j^T A H_j$
  end
end

TABLE 2.7
Rowwise reduction to J–Hessenberg Form

REMARK 2.23. *As in the SR algorithm only symplectic similarity transformations are employed, it preserves the Hamiltonian structure. That is, if A is Hamiltonian, then all iterates A_i of the SR algorithm are Hamiltonian.*

2.2.3. HR **Algorithm.** The HR algorithm [32, 35] is just like the SR algorithm a member of the family of GR algorithms [142] for calculating eigenvalues and invariant subspaces of matrices. Unlike the SR algorithm, the HR algorithm deals with matrices in $\mathbb{R}^{n\times n}$. Before we briefly introduce the HR algorithm, we recall some definitions from Definition 2.2. A signature matrix is a diagonal matrix $D = \mathrm{diag}(d_1, \ldots, d_n)$ such that each $d_i \in \{1, -1\}$. Given a signature matrix D, we say that a matrix $A \in \mathbb{R}^{n\times n}$ is D–symmetric if $(DA)^T = DA$. Moreover, from Lemma 2.3 we know that a tridiagonal matrix T is D–symmetric for some D if and only if $|t_{i+1,i}| = |t_{i,i+1}|$ for $i = 1, \ldots, n-1$. Every unreduced tridiagonal matrix is similar to

a D–symmetric matrix (for some D) by a diagonal similarity with positive main diagonal entries. D–symmetric tridiagonal matrices are generated by the nonsymmetric Lanczos process [82].

Almost every $A \in \mathbb{R}^{n\times n}$ has an HR decomposition

$$A = HU,$$

in which U is upper triangular, and H satisfies the hyperbolic property

$$H^T DH = \widehat{D},$$

where $\widehat{D}$ is another signature matrix [35]. For nonsingular A the HR decomposition is unique up to a signature matrix. We can make it unique by insisting that the upper triangular factor U satisfies $u_{ii} > 0$, $i = 1, \ldots, n$. The HR algorithm [32, 35] is an iterative process based on the HR decomposition. Choose a spectral transformation function p for which $p(A)$ is nonsingular, and form the HR decomposition of $p(A)$, if possible:

$$p(A) = HU.$$

Then use H to perform a similarity transformation on A to get the next iterate:

$$\widehat{A} = H^{-1}AH.$$

The HR algorithm has the following structure preservation property: If A is D–symmetric and $H^T DH = \widehat{D}$, then $\widehat{A}$ is $\widehat{D}$–symmetric. If A is also tridiagonal, then so is $\widehat{A}$. For a detailed discussion see [35, 32]. See also [34, 93].

At the end of Section 4.1, we will see that there is a close connection between the SR algorithm on $2n \times 2n$ Hamiltonian matrices and the HR algorithm on $n \times n$ tridiagonal D-symmetric matrices.

2.3. Lanczos Algorithm. In 1950, Lanczos [82] proposed a method for the successive reduction of a given, in general non-Hermitian, $n\times n$ matrix A to tridiagonal form. The Lanczos process generates a sequence $T^{m,m}, m = 1, 2, ...,$ of $m\times m$ matrices which, in a certain sense, approximate A. Furthermore, in exact arithmetic and if no breakdown occurs, the Lanczos method terminates after at most $l \leq n$ steps with $T^{l,l}$ a tridiagonal matrix that represents the restriction of A or A^T to an A–invariant or A^T–invariant subspace of $\mathbb{C}^n$, respectively. In particular, all the eigenvalues of $T^{l,l}$ are also eigenvalues of A and, in addition, the method produces basis vectors for the A–invariant or A^T–invariant subspace found.

The nonsymmetric Lanczos tridiagonalization is essentially the Gram-Schmidt bi-orthogonalization method for generating bi-orthogonal bases for a pair of Krylov subspaces

$$\{q, Aq, A^2q, A^3q, ...\} \quad \text{and} \quad \{v^T, v^TA, v^TA^2, v^TA^3, ...\}$$

($v, q \in \mathbb{R}^n$ arbitrary). Applying the two-sided Gram-Schmidt process to the vectors $\{A^kq\}_{k\geq 0}$ and $\{v^TA^k\}_{k\geq 0}$, one arrives at a three term recurrence relation which, when $k = n$, represents a similarity transformation of the matrix A to tridiagonal form. The three term recurrence relation produces a sequence of vectors which can be viewed as forming the rows and columns, respectively, of rectangular matrices, $V^{n,k}$ and $Q^{n,k}$, such that after n steps, $V^{n,n}$ and $Q^{n,n}$ are $n \times n$ matrices with $Q^{n,n} = (V^{n,n})^{-1}$ and

$V^{n,n}AQ^{n,n}$ is tridiagonal. At each step, an orthogonalization is performed, which requires a division by the inner product of (multiples of) the vectors produced at the previous step. Thus the algorithm suffers from breakdown and instability if any of these inner products is zero or nearly zero.

Let us be more precise. Given $v_1, q_1 \in \mathbb{R}^n$ and a nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$, the standard nonsymmetric Lanczos algorithm produces matrices $V^{n,k} = [v_1, \ldots, v_k] \in \mathbb{R}^{n\times k}$ and $Q^{n,k} = [q_1, \ldots, q_k] \in \mathbb{R}^{n\times k}$ which satisfy the recursive identities

$$AV^{n,k} = V^{n,k}T^{k,k} + \beta_{k+1}v_{k+1}e_k^T, \tag{2.8}$$
$$A^TQ^{n,k} = Q^{n,k}(T^{k,k})^T + \gamma_{k+1}q_{k+1}e_k^T, \tag{2.9}$$

where

$$T^{k,k} = \begin{bmatrix} \alpha_1 & \gamma_2 & & \\ \beta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_k \\ & & \beta_k & \alpha_k \end{bmatrix}$$

is a truncated reduction of A. Generally, the elements β_j and γ_j are chosen so that $|\beta_j| = |\gamma_j|$ and $(Q^{n,k})^TV^{n,k} = I^{k,k}$ (bi-orthogonality). One pleasing result of this bi-orthogonality condition is that multiplying (2.8) on the left by $(Q^{n,k})^T$ yields the relationship $(Q^{n,k})^TAV^{n,k} = T^{k,k}$.

Encountering a zero $\beta_{k+1}v_{k+1}$ or $\gamma_{k+1}q_{k+1}$ in the Lanczos iteration is a welcome event in that it signals the computation of an exact invariant subspace. If $\beta_{k+1}v_{k+1} = 0$, then the iteration terminates and $\text{span}\{v_1, \ldots, v_k\}$ is an invariant subspace for A. If $\gamma_{k+1}q_{k+1} = 0$, then the iteration also terminates and $\text{span}\{q_1, \ldots, q_k\}$ is an invariant subspace for A^T. If neither condition is true and $q_{k+1}^Tv_{k+1} = 0$, then the tridiagonalization process ends without any invariant subspace information. This is called a serious breakdown. However, an exact zero or even a small $\beta_{k+1}v_{k+1}$ or $\gamma_{k+1}q_{k+1}$ is a rarity in practice. Nevertheless, the extremal eigenvalues of $T^{k,k}$ turn out to be surprisingly good approximations to A's extremal eigenvalues. Hence, the successive tridiagonalization by the Lanczos algorithm combined with a suitable method for computing the eigenvalues of the resulting tridiagonal matrices is an appropriate iterative method for solving large and sparse eigenproblem, if only some of the eigenvalues are sought. As the resulting tridiagonal matrices are sign-symmetric, the HR or the LR algorithm are appropriate QR like methods for computing the eigenvalues and invariant subspaces, as they preserve the special structure.

Yet, in practice, there are a number of difficulties associated with the Lanczos algorithm. At each step of the nonsymmetric Lanczos tridiagonalization, an orthogonalization is performed, which requires a division by the inner product of (multiples of) the vectors produced at the previous step. Thus the algorithm suffers from breakdown and instability if any of these inner products is zero or close to zero. It is known [71] that vectors q_1 and v_1 exist so that the Lanczos process with these as starting vectors does not encounter breakdown. However, determining these vectors requires knowledge of the minimal polynomial of A. Further, there are no theoretical results showing that v_1 and q_1 can be chosen such that small inner products can be avoided. Thus, no algorithm for successfully choosing v_1 and q_1 at the start of the computation yet exists.

In theory, the three-term recurrences in (2.8) and (2.9) are sufficient to guarantee $(Q^{n,k})^T V^{n,k} = I^{k,k}$. It is known [106] that bi-orthogonality will in fact be lost when at least one of the eigenvalues of $T^{k,k}$ converges to an eigenvalue of A. In order to overcome this problem, re-bi-orthogonalization of the vectors q_j and v_j is necessary. Different strategies have been proposed for this, e.g., complete or selective re-bi-orthogonalization. For a more detailed discussion on the various aspects of the difficulties of the Lanczos method in the context of computing some eigenvalues of large and sparse matrices see, e.g., [120] and the references therein.

It is possible to modify the Lanczos process so that it skips over exact breakdowns. A complete treatment of the modified Lanczos algorithm and its intimate connection with orthogonal polynomials and Padé approximation was presented by Gutknecht [69, 70]. Taylor [133] and Parlett, Taylor, and Liu [112] were the first to propose a look-ahead Lanczos algorithm that skips over breakdowns and near-breakdowns. The price paid is that the resulting matrix is no longer tridiagonal, but has a small bulge in the tridiagonal form to mark each occurrence of a (near) breakdown. Freund, Gutknecht, and Nachtigal presented in [63] a look-ahead Lanczos code that can handle look-ahead steps of any length.

A different approach to deal with the inherent difficulties of the Lanczos process is to modify the starting vectors by an implicitly restarted Lanczos process (see the fundamental work in [46, 126]; for the nonsymmetric eigenproblem the implicitly restarted Arnoldi method has been implemented very successfully, see [90]). The problems are addressed by fixing the number of steps in the Lanczos process at a prescribed value k which is dependent on the required number of approximate eigenvalues. J–orthogonality of the k Lanczos vectors is secured by re–J–orthogonalizing these vectors when necessary. The purpose of the implicit restart is to determine initial vectors such that the associated residual vectors are tiny. Given that $V^{n,k}$ and $Q^{n,k}$ from (2.8) and (2.9) are known, an implicit Lanczos restart computes the Lanczos factorization

$$A\widetilde{V}^{n,k} = \widetilde{V}^{n,k}\widetilde{T}^{k,k} + \widetilde{r}_k e_k^T, \tag{2.10}$$

$$A^T\widetilde{Q}^{n,k} = \widetilde{Q}^{n,k}(\widetilde{T}^{k,k})^T + \widetilde{q}_k e_k^T, \tag{2.11}$$

which corresponds to the starting vectors

$$\widetilde{v}_1 = \rho_p(A - \mu I)v_1, \qquad \widetilde{q}_1 = \rho_q(A^T - \mu I)q_1, \tag{2.12}$$

without explicitly restarting the Lanczos process with the vectors in (2.12). For a detailed derivation see [66] and the related work in [46, 126].

For nonsymmetric problems, convergence of Krylov projection methods has been studied extensively. Saad [119] developed a bound for matrices with simple eigenvalues for the gap between a single eigenvector and the Krylov subspace. This result was generalized in [72] to include defective matrices, but the bounds explicitly involve the Jordan canonical form and derivatives of approximating polynomials. Simoncini [124] analyses convergence of a block Arnoldi method for defective matrices using pseudo-spectra. Lehoucq [88] relates the implicitly restarted Arnoldi method to subspace iteration to analyze convergence to an invariant subspace. Calvetti, Reichel and Sorensen [46] introduce concepts from potential theory to analyse the convergence in the implicitly restarted Lanczos algorithm to a single eigenvector for Hermitian matrices. In the nonsymmetric case, the possibility of nonnormality complicates the analysis considerable. The possibility of derogatory matrices my even render certain

invariant subspaces unreachable. These concepts are introduced by Beatti, Embree and Rossi [11]. They employ various idea from functional analysis, pseudo-spectra and potential theory. Their analysis focuses on convergence in gap of a (restarted) Krylov space to a desired invariant subspace, and they are able to treat convergence in full generality. For a very nice sketch of their main ideas in a comprehensive way that can be readily understood see [128].

3. Reduction to Hamiltonian J-Hessenberg form and the parameterization of a Hamiltonian matrix. Applying the algorithm for reducing a matrix to J-Hessenberg form to a Hamiltonian matrix H (see Table 2.6), a symplectic transformation matrix S is computed such that

$$SHS^{-1} = \begin{bmatrix} \diagdown & \diagdown \\ \diagdown & \diagdown \end{bmatrix} = \begin{bmatrix} \widehat{A} & \widehat{G} \\ \widehat{Q} & -\widehat{A}^T \end{bmatrix},$$

where $\widehat{Q} = \widehat{Q}^T, \widehat{G} = \widehat{G}^T$ as the resulting matrix has to be of Hamiltonian form again. Hence, $\widehat{A}$ and $\widehat{Q}$ have to be diagonal matrices, $\widehat{G}$ a symmetric tridiagonal matrix. That is, the reduced Hamiltonian matrix is of the form

$$\left[\begin{array}{ccccc|ccccc} \delta_1 & & & & & \beta_1 & \zeta_2 & & & \\ & \delta_2 & & & & \zeta_2 & \beta_2 & \ddots & & \\ & & \ddots & & & & \ddots & \ddots & \zeta_n & \\ & & & \delta_n & & & & \zeta_n & \beta_n & \\ \hline \nu_1 & & & & & -\delta_1 & & & & \\ & \nu_2 & & & & & -\delta_2 & & & \\ & & \ddots & & & & & \ddots & & \\ & & & \nu_n & & & & & -\delta_n & \end{array}\right]. \tag{3.1}$$

Hence any $2n \times 2n$ Hamiltonian matrix is symplectically similar to a Hamiltonian J-Hessenberg matrix which is determined by $4n-1$ parameters. A Hamiltonian J-Hessenberg matrix will be called unreduced, if $\nu_j \neq 0$ for all $j = 1, \ldots, n$ and $\zeta_k \neq 0$ for all $k = 2, \ldots, n$. This definition is in analogy to the definition of an unreduced Hessenberg matrix, as

$$PHP^T = \left[\begin{array}{cc|cc|cc|cc} \delta_1 & \beta_1 & 0 & \zeta_2 & & & & \\ \nu_1 & -\delta_1 & 0 & 0 & & & & \\ \hline 0 & \zeta_2 & \delta_2 & \beta_2 & \cdot & & & \\ 0 & 0 & \nu_2 & -\delta_2 & & \cdot & & \\ \hline & & \cdot & & \cdot & & 0 & \zeta_n \\ & & & \cdot & & \cdot & 0 & 0 \\ \hline & & & & 0 & \zeta_n & \delta_n & \beta_n \\ & & & & 0 & 0 & \nu_n & -\delta_n \end{array}\right], \tag{3.2}$$

where P is as in (2.2). Unreduced J-Hessenberg matrices have similar properties as unreduced Hessenberg matrices, e.g., Theorem 2.13 on the eigenvalues and eigenvectors holds.

As we will make use of the reduction to the Hamiltonian J-Hessenberg form in the following, let us have a closer look at the algorithm. We illustrate the major

elimination steps on a 6×6 example. Let

$$H = \left[\begin{array}{ccc|ccc} x & x & x & a & b & c \\ x & x & x & b & d & e \\ x & x & x & c & e & f \\ \hline m & p & q & x & x & x \\ p & r & s & x & x & x \\ q & s & t & x & x & x \end{array}\right] = \left[\begin{array}{cc} A & G \\ Q & -A^T \end{array}\right]$$

be a Hamiltonian matrix. In the first step of the reduction to J-Hessenberg form, the entries below the diagonal element in the first column of Q are eliminated by a series of symplectic Givens transformations. Due to the Hamiltonian structure of H which is kept in the course of the computations, simultaneously the corresponding elements in the first row of Q are annihilated. For our example, this yields

$$\left[\begin{array}{ccc|ccc} x & x & x & a & b & c \\ x & x & x & b & d & e \\ x & x & x & c & e & f \\ \hline m & 0 & 0 & x & x & x \\ 0 & r & s & x & x & x \\ 0 & s & t & x & x & x \end{array}\right]$$

Next a symplectic Householder matrix is used to reduce the rest of the first column as far as possible without destroying the already achieved zeros;

$$\left[\begin{array}{ccc|ccc} x & x & x & a & b & c \\ x & x & x & b & d & e \\ 0 & x & x & c & e & f \\ \hline m & 0 & 0 & x & x & 0 \\ 0 & r & s & x & x & x \\ 0 & s & t & x & x & x \end{array}\right].$$

Due to the Hamiltonian structure of the matrix, this creates additional zeros in the $(2,2)$-block. A symplectic Gauss transformation completes the reduction of the first column

$$\left[\begin{array}{ccc|ccc} x & x & x & a & b & c \\ 0 & x & x & b & d & e \\ 0 & x & x & c & e & f \\ \hline m & 0 & 0 & x & 0 & 0 \\ 0 & r & s & x & x & x \\ 0 & s & t & x & x & x \end{array}\right].$$

As before, due to the Hamiltonian structure of the matrix, this creates additional zeros in the $(2,2)$-block. Next, the first columns/rows of the blocks G and $-A^T$ are treated. First a series of symplectic Givens transformations are used in order to eliminate the entries below the diagonal element in the first column of the current $(2,2)$-block. Due to the Hamiltonian structure of the matrix, this creates additional

zeros in the $(1,1)$-block.

$$\left[\begin{array}{ccc|ccc} x & 0 & 0 & a & b & c \\ 0 & x & x & b & d & e \\ 0 & x & x & c & e & f \\ \hline m & 0 & 0 & x & 0 & 0 \\ 0 & r & s & 0 & x & x \\ 0 & s & t & 0 & x & x \end{array}\right].$$

The first reduction step is completed by a symplectic Householder transformation which brings the first column/row of the current G in the desired form

$$\left[\begin{array}{ccc|ccc} x & 0 & 0 & a & b & 0 \\ 0 & x & x & b & d & e \\ 0 & x & x & 0 & e & f \\ \hline m & 0 & 0 & x & 0 & 0 \\ 0 & r & s & 0 & x & x \\ 0 & s & t & 0 & x & x \end{array}\right].$$

Next the second column/row of the 4 blocks are treated bringing our 6×6 matrix into the desired Hamiltonian J-Hessenberg form.

3.1. A canonical form for Hamiltonian J-Hessenberg matrices . We have noted that the Hamiltonian J-Hessenberg form is preserved by the SR algorithm. The outcome of an SR iteration is not quite uniquely determined; it is determined up to a similarity transformation by a trivial matrix (2.5). It is therefore of interest to develop a canonical form for Hamiltonian J-Hessenberg matrices under similarity transformations by trivial matrices. We restrict our attention to unreduced Hamiltonian J-Hessenberg matrices, since every Hamiltonian J-Hessenberg matrix can be decomposed into two or more smaller unreduced ones.

THEOREM 3.1. *Let $\widetilde{H}$ be an unreduced Hamiltonian J-Hessenberg matrix. Then there exists a symplectic J-triangular matrix X such that $H = X^{-1}\widetilde{H}X$ has the canonical form*

$$H = \left[\begin{array}{cc} 0 & V \\ D & 0 \end{array}\right], \tag{3.3}$$

where D is a signature matrix, and V is a symmetric, irreducible tridiagonal matrix. D is uniquely determined, V is determined by to a similarity transformation by a signature matrix, and X is unique up to multiplication by a signature matrix of the form $\mathrm{diag}(C, C)$. *Let T denote the D-symmetric matrix DV. The eigenvalues of T are $\lambda_i^2, i = 1, \ldots, n$, where $\lambda_i, -\lambda_i$ are the eigenvalues of H.*

Proof. See [21]. √

REMARK 3.2. *(from [21])*

- *The canonical form could be made unique by insisting that either T's or V's subdiagonal entries be positive.*
- *From the standpoint of numerical stability, it might not be advisable to transform a Hamiltonian matrix into canonical form. In the process, the spectral information $\pm\lambda$ is condensed into T as λ^2. Any small eigenvalues of $\widetilde{H}$ are transformed to tiny eigenvalues of T, which are then extremely vulnerable to roundoff errors in any subsequent computations on T.*

- *We note that $H^2 = \mathrm{diag}(T^T, T)$. Thus, forming T is tantamount to squaring H. Squaring a Hamiltonian matrix to compute its eigenvalues is also the basis of Van Loan's square reduced method [136]. An error estimate for retrieving the eigenvalues of a Hamiltonian matrix form their squares computed by Van Loan's method is given in [136] and indicates that one may loose up to half the significant digits as compared to a numerically backward stable method as the QR method. The same limitations in accuracy apply if we transform a Hamiltonian J-Hessenberg matrix into canonical form and compute its eigenvalues via T.*
- *The eigenvectors of H can be recovered from those of T. If $Ty = \lambda^2 y$, $(y \neq 0)$, then $[\pm(Dy)^T, y^T]$ are eigenvectors of H associated with eigenvalues $\pm\lambda$.*

4. The Hamiltonian SR algorithm. Eigenvalues and eigenvectors of Hamiltonian J-Hessenberg matrices can be computed efficiently by the SR algorithm. This has already been discussed to some extent in [38, 17, 16, 145]. If H is the current iterate, then a spectral transformation function q is chosen (such that $q(H) \in \mathbb{R}^{2n\times 2n}$) and the SR decomposition of $q(H)$ is formed, if possible:

$$q(H) = SR.$$

Then the symplectic factor S is used to perform a similarity transformation on H to yield the next iterate, which we will call $\widehat{H}$;

$$\widehat{H} = S^{-1}HS. \tag{4.1}$$

If $\mathrm{rank}(q(H)) = 2n$ and H is a Hamiltonian J-Hessenberg matrix, then so is $\widehat{H}$. If $\mathrm{rank}(q(H)) = 2n-\nu =: 2k$ and H is an unreduced Hamiltonian J-Hessenberg matrix, then $\widehat{H}$ in (4.1) is of the form

$$\widehat{H} = \left[\begin{array}{cc|cc} \ddots & & \ddots & \\ & \square & & \square \\ \hline \ddots & & \ddots & \\ & \square & & \square \end{array}\right] = \left[\begin{array}{cc|cc} \widehat{H}_{11} & & \widehat{H}_{13} & \\ & \widehat{H}_{22} & & \widehat{H}_{24} \\ \hline \widehat{H}_{31} & & -\widehat{H}_{11}^T & \\ & \widehat{H}_{42} & & -\widehat{H}_{22}^T \end{array}\right], \tag{4.2}$$

where $\widehat{H}_{11}, \widehat{H}_{13}, \widehat{H}_{31} \in \mathbb{R}^{k\times k}$ and $\widehat{H}_{22}, \widehat{H}_{24}, \widehat{H}_{42} \in \mathbb{R}^{n-k\times n-k}$ and

- $\left[\begin{array}{cc} \widehat{H}_{11} & \widehat{H}_{13} \\ \widehat{H}_{31} & -\widehat{H}_{11}^T \end{array}\right]$ is a Hamiltonian J-Hessenberg matrix,
- the eigenvalues of $\left[\begin{array}{cc} \widehat{H}_{22} & \widehat{H}_{24} \\ \widehat{H}_{42} & -\widehat{H}_{22}^T \end{array}\right]$ are the ν roots of $q(H)$ that are eigenvalues of H.

For a proof see [141, Theorem 4.5], see also [18, Theorem 4.1].

An algorithm for computing S and R explicitly is presented in [38]. As with explicit QR steps, the expense of explicit SR steps comes from the fact that $q(H)$ has to be computed explicitly. A preferred alternative is the implicit SR step, an analogue to the Francis QR step [59]. The first implicit transformation S_1 is selected

in order to introduce a bulge into the J-Hessenberg matrix H. That is, a symplectic matrix S_1 is determined such that

$$S_1^{-1} q(H) e_1 = \alpha e_1, \qquad \alpha \in \mathbb{R},$$

where $q(H)$ is an appropriately chosen spectral transformation function. Applying this first transformation to the J-Hessenberg matrix yields a Hamiltonian matrix $S_1^{-1} H S_1$ with almost J-Hessenberg form having a small bulge. The remaining implicit transformations perform a bulge-chasing sweep down the subdiagonals to restore the J-Hessenberg form. That is, a symplectic matrix S_2 is determined such that $S_2^{-1} S_1^{-1} H S_1 S_2$ is of J-Hessenberg form again. If H is an unreduced J-Hessenberg matrix and $\operatorname{rank}(q(H)) = 2n$, then $\widetilde{H} = S_2^{-1} S_1^{-1} H S_1 S_2$ is also an unreduced J-Hessenberg matrix. Hence, there will be parameters $\widetilde{\delta}_1, \ldots, \widetilde{\delta}_n, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_n, \widetilde{\zeta}_1, \ldots, \widetilde{\zeta}_n$, $\widetilde{\nu}_2, \ldots, \widetilde{\nu}_n$ which determine $\widetilde{H}$. The algorithm for reducing a matrix to J-Hessenberg form as given in Table 2.6 can be used as a building block for the implicit SR step. An efficient implementation of the SR step for Hamiltonian J-Hessenberg matrices as it will be derived in the next section involves $\mathcal{O}(n)$ arithmetic operations. Hence a gain in efficiency is obtained compared to the SR algorithm on J-Hessenberg matrices where each SR step involves $\mathcal{O}(n^2)$ arithmetic operations.

Due to the special Hamiltonian eigenstructure, the spectral transformation function will be chosen either as

$$q_2(H) = (H - \mu I)(H + \mu I), \qquad \mu \in \mathbb{R} \ \text{ or } \ \mu = i\omega, \omega \in \mathbb{R},$$

or

$$q_4(H) = (H - \mu I)(H + \mu I)(H - \overline{\mu} I)(H + \overline{\mu} I), \qquad \mu \in \mathbb{C}, \operatorname{Re}(\mu) \neq 0.$$

If the chosen shifts are good approximate eigenvalues, we expect deflation at the end of the SR step as indicated in (4.2). As proposed in [38], a shift strategy similar to that used in the standard QR algorithm should be used. For example, for a quadruple shift, we choose the 4 eigenvalues of the 4×4 Hamiltonian J-Hessenberg submatrix

$$H_{4\times 4} = \left[\begin{array}{cc|cc} \delta_{n-1} & & \beta_{n-1} & \zeta_n \\ & \delta_n & \zeta_n & \beta_n \\ \hline \nu_{n-1} & & -\delta_{n-1} & \\ & \nu_n & & -\delta_n \end{array}\right].$$

There is no need to compute the eigenvalues of $H_{4\times 4}$ directly. Comparing $q_4(H)$

$$q_4(H) = H^4 - (\mu^2 + \overline{\mu}^2) H^2 + \mu^2 \overline{\mu}^2 I$$

with the characteristic polynomial of $H_{4\times 4}$

$$\det(H_{4\times 4} - \lambda I) = \lambda^4 - \lambda^2 (a_{n-1} + a_n) + a_{n-1} a_n - \nu_{n-1} \nu_n \zeta_n^2,$$

where

$$a_k = \delta_k^2 + \nu_k \beta_k,$$

we obtain

$$q_4(H) = H^4 - (a_{n-1} + a_n) H^2 + (a_{n-1} a_n - \nu_{n-1} \nu_n \zeta_n^2) I.$$

The first column of $q_4(H)$ which is needed to start the implicit SR step is then given by

$$\begin{aligned} q_4(H)e_1 = (a_1^2 + \zeta_2^2\nu_1\nu_2 - (a_{n-1}+a_n)a_1 + a_{n-1}a_n - \nu_{n-1}\nu_n\zeta_n^2)e_1 \\ + \zeta_2\nu_1[(a_1+a_2) - (a_{n-1}+a_n)]e_2 + \nu_1\nu_2\zeta_2\zeta_3 e_3. \end{aligned} \tag{4.3}$$

This is exactly the generalized Rayleigh-quotient strategy for choosing the shifts proposed by Watkins and Elsner in [142]. Hence the convergence theorems Theorem 6.2, 6.3 and 6.5 from [142] (see also Theorem 2.20 and Theorem 2.22 in Section 2.2.2) can be applied here. In particular the Hamiltonian SR algorithm is typically convergent. Let $H_0 \in \mathbb{R}^{2n\times 2n}$ have distinct eigenvalues. Let (H_i) be the sequence generated by the SR algorithm starting from H_0, using the generalized Rayleigh-quotient shift strategy with polynomials of degree 4. Each of the iterates

$$PH_iP^T = \begin{bmatrix} X_{11}^{(i)} & X_{12}^{(i)} \\ X_{21}^{(i)} & X_{22}^{(i)} \end{bmatrix}, X_{22}^{(i)} \in \mathbb{R}^{4\times 4}$$

satisfies $||X_{12}^{(i)}|| = ||X_{21}^{(i)}||$ for some fixed norm $||\cdot||$, as

$$X_{21}^{(i)} = \left[\begin{array}{cc|c|cc} 0 & 0 & \cdots & 0 & \zeta_{n-1} \\ 0 & 0 & \cdots & 0 & 0 \\ \hline 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{array}\right], \qquad X_{12}^{(i)} = \left[\begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline & \vdots & & \vdots \\ \hline 0 & \zeta_{n-1} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right].$$

Then, from [142, Theorem 6.5] (see also Theorem 2.22), the iterates converge cubically if they converge.

By applying a sequence of quadruple shift SR steps to a Hamiltonian J-Hessenberg matrix H it is possible to reduce the tridiagonal block in H to quasi-diagonal form with 1×1 and 2×2 blocks on the diagonal. The eigenproblem decouples into a number of simple Hamiltonian 2×2 or 4×4 eigenproblems. In doing so, it is necessary to monitor the off-diagonal elements in the tridiagonal block of H in order to bring about decoupling whenever possible. Decoupling occurs if $\zeta_j = 0$ for some j as the $(1,2)$ block of H decouples

$$\left[\begin{array}{ccccc|ccccc} \beta_1 & \zeta_2 & & & & & & & & \\ \zeta_2 & \ddots & \ddots & & & & & & & \\ & \ddots & \beta_{j-2} & \zeta_{j-1} & & & & & & \\ & & \zeta_{j-1} & \beta_{j-1} & 0 & & & & & \\ \hline & & & 0 & \beta_j & \zeta_{j+1} & & & & \\ & & & & \zeta_{j+1} & \beta_{j+1} & \ddots & & & \\ & & & & & \ddots & \ddots & \zeta_n & & \\ & & & & & & \zeta_n & \beta_n & & \end{array}\right].$$

When dealing with upper Hessenberg matrices, as in the QR setting, decoupling occurs whenever a subdiagonal element becomes zero. In practice, decoupling is said to occur whenever a subdiagonal element in the Hessenberg matrix X is suitably small. For example, in LAPACK [4] if

$$|x_{p+1,p}| \le c\mathbf{u}(|x_{pp}| + |x_{p+1,p+1}|)$$

for some small constant c and the unit roundoff $\mathbf{u}$, then $x_{p+1,p}$ is declared to be zero. This is justified since rounding errors of order $\mathbf{u}||X||$ are already present throughout the matrix.

Taking the same approach here and recalling that a perfect shuffle of all rows and columns of a Hamiltonian J-Hessenberg matrix yields a standard upper Hessenberg matrix (3.2), we check whether

$$|\zeta_j| \leq \epsilon(|\delta_{j-1}| + |\delta_j|)$$

is satisfied; in this case we will have deflation. Here ϵ is a small constant, e.g., $\epsilon = c\mathbf{u}$.

REMARK 4.1. *As mentioned in (3.2), performing a perfect shuffle on a Hamiltonian J-Hessenberg matrix H yields an upper Hessenberg matrix*

$$PHP^T = \left[\begin{array}{cc|cc|cc|cc} \delta_1 & \beta_1 & 0 & \zeta_2 & & & & \\ \nu_1 & -\delta_1 & 0 & 0 & & & & \\ \hline 0 & \zeta_2 & \delta_2 & \beta_2 & \cdot & & & \\ 0 & 0 & \nu_2 & -\delta_2 & & \cdot & & \\ \hline & & \cdot & & \cdot & & 0 & \zeta_n \\ & & & \cdot & & \cdot & 0 & 0 \\ \hline & & & & 0 & \zeta_n & \delta_n & \beta_n \\ & & & & 0 & 0 & \nu_n & -\delta_n \end{array}\right].$$

In case $\zeta_k = 0$, deflation in the usual sense takes place as discussed above. In case a ν_k becomes zero in the course of the iteration, two eigenvalues and one right and one left eigenvector can be read off in such a case.

We proceed the process of applying quadruple SR steps to a Hamiltonian J-Hessenberg matrix until the problem has completely split into subproblems of dimension 2 or 4, see Table 4.1 (in MATLAB-like notation). In a final step we then have to solve these small subproblems in order to compute a real Schur-like form from which eigenvalues and invariant subspaces can be read off. The details for this computation are given in Section 6.

4.1. Equivalence of the HR and the Hamiltonian SR algorithm. Consider an SR iteration on a Hamiltonian matrix H. Since the eigenvalues occur in plus-minus pairs, it is reasonable to choose the shifts in plus-minus pairs. If we wish to effect an SR iteration of degree $2k$ with shifts $\pm\mu_i, i = 1, \ldots, k$, we use the polynomial

$$q(H) = \prod_{i=1}^{k}(H - \mu_i I)(H + \mu_i I) = \prod_{i=1}^{k}(H^2 - \mu_i^2 I).$$

In order to simplify the discussion, we assume that $q(H)$ is nonsingular. Nothing bad happens in the singular case [141]. We also insist that complex shifts be present in conjugate pairs, so that $q(H)$ is real.

THEOREM 4.2. *Let*

$$H = \begin{bmatrix} 0 & V \\ D & 0 \end{bmatrix}$$

be an unreduced Hamiltonian J-Hessenberg matrix in canonical form (3.3). Then an SR algorithm of degree $2k$ with shifts $\pm\mu_i, i = 1, \ldots, k$ on H is equivalent to an HR iteration of degree k with shifts $\mu_i^2, i = 1, \ldots, k$ on the D-symmetric matrix $T = DV$.

Algorithm: SR Iteration

Given a $2n \times 2n$ Hamiltonian J-Hessenberg matrix H compute a $2n \times 2n$ symplectic matrix S such that $\widehat{H} = S^{-1}HS$ is a Hamiltonian J-Hessenberg matrix which the $(1,1), (1,2), (2,1)$ and the $(2,2)$ blocks are each block-diagonal where all blocks are either 1×1 or 2×2. Moreover, the block structure for all four blocks of $\widehat{H}$ is the same. Thus the eigenproblem for $\widehat{H}$ decouples into 2×2 and 4×4 subproblems.

$q = 0$
repeat until $q = n$
set all ζ_i to zero that satisfy $|\zeta_i| \leq \epsilon(|\delta_{i-1}| + |\delta_i|)$
find the largest nonnegative q and the smallest nonnegative p
such that if

$$H = \left[\begin{array}{ccc|ccc} A_{11} & & & G_{11} & & \\ & A_{22} & & & G_{22} & \\ & & A_{33} & & & G_{33} \\ \hline Q_{11} & & & -A_{11}^T & & \\ & Q_{22} & & & -A_{22}^T & \\ & & Q_{33} & & & -A_{33}^T \end{array}\right],$$

where $T_{11} \in \mathbb{R}^{p \times p}, T_{22} \in \mathbb{R}^{(n-p-q)\times(n-p-q)}$, and $T_{33} \in \mathbb{R}^{q \times q}$ for $T \in \{A, G, Q\}$, then G_{33} is block diagonal with 1×1 and 2×2 blocks and G_{22} is unreduced symmetric tridiagonal.
if $q < n$
perform a quadruple SR step on

$$\left[\begin{array}{c|c} A_{22} & G_{22} \\ \hline Q_{22} & -A_{22}^T \end{array}\right]$$

update H accordingly
end
end

TABLE 4.1
SR Iteration

Proof. See [21]. √

Please note, that this result is of theoretical interest only, as the resulting method suffers from a possible loss of half the significant digits during the transformation to canonical form.

5. The parameterized SR step. As will be shown in this section, the SR algorithm for a Hamiltonian J-Hessenberg matrix H can be rewritten in a parameterized form that will work only with the $4n - 1$ parameters which determine H instead of the entire matrix in each iteration step. Thus only $\mathcal{O}(n)$ flops per SR step are needed compared to $\mathcal{O}(n^3)$ flops when working on the actual Hamiltonian matrix. The key to the development of a SR algorithm working only on the parameters is the observation

that at any point in the implicit SR step only a certain, limited number of rows and columns of the Hamiltonian J-Hessenberg matrix is worked on. In the leading part of the intermediate matrices the Hamiltonian J-Hessenberg form is already retained and is not changed any longer, while the trailing part has not been changed yet. Hence, from the leading part the first parameters of the resulting J-Hessenberg matrix can be read off, while from the trailing part the last parameters of the original J-Hessenberg matrix can still be read off. Recall the implicit SR step as described in Section 2. The first implicit transformation S_1 is selected in order to introduce a bulge into the J-Hessenberg matrix H. That is, a symplectic matrix S_1 is determined such that

$$S_1^{-1} q(H) e_1 = \alpha e_1, \qquad \alpha \in \mathbb{R},$$

where $q(H)$ is an appropriately chosen spectral transformation function. Applying this first transformation to the J-Hessenberg matrix yields a Hamiltonian matrix $S_1^{-1} H S_1$ with almost J-Hessenberg form having a small bulge. The remaining implicit transformations perform a bulge-chasing sweep down the subdiagonals to restore the J-Hessenberg form. That is, a symplectic matrix S_2 is determined such that $S_2^{-1} S_1^{-1} H S_1 S_2$ is of J-Hessenberg form again. If H is an unreduced J-Hessenberg matrix and $\operatorname{rank}(q(H)) = 2n$, then $\widetilde{H} = S_2^{-1} S_1^{-1} H S_1 S_2$ is also an unreduced J-Hessenberg matrix. Hence, there will be parameters $\widetilde{\delta}_1, \ldots, \widetilde{\delta}_n, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_n, \widetilde{\zeta}_1, \ldots, \widetilde{\zeta}_n$, $\widetilde{\nu}_2, \ldots, \widetilde{\nu}_n$ which determine $\widetilde{H}$. During the bulge-chasing sweep the bulge is successively moved down the subdiagonals, one row and one column at a time. Our goal in this section will be to derive these parameters directly from the original ones.

Due to the special Hamiltonian eigenstructure, the spectral transformation function will be chosen either as

$$q_2(H) = (H - \mu I)(H + \mu I), \qquad \mu \in \mathbb{R} \ \text{ or } \ \mu = i\omega, \omega \in \mathbb{R},$$

or

$$q_4(H) = (H - \mu I)(H + \mu I)(H - \overline{\mu} I)(H + \overline{\mu} I), \qquad \mu \in \mathbb{C}, \operatorname{Re}(\mu) \neq 0.$$

In case an exceptional shift step is needed in the SR algorithm, one might want to use a single shift

$$q_1(H) = H - \mu I, \qquad \mu \in \mathbb{R}.$$

We will consider each of the three cases separately.

5.1. A single shift implicit SR step. Consider a single shift implicit SR step, which might be used as an exceptional shift step in the SR iteration. As H is a real matrix, for a single shift the shift polynomial $q_1(H) = H - \mu I$ should be chosen for $\mu \in \mathbb{R}$. The first column of q_1 is of the form

$$x = q_1(H) e_1 = (\delta_1 - \mu) e_1 + \nu_1 e_{n+1}.$$

This vector can be transformed into a multiple of e_1 by a symplectic Givens transformation G_1 where the parameters c_1, s_1 are given by $[c_1, s_1] = \mathbf{givens}(\delta_1 - \mu, \nu_1)$. Hence, the first step of the implicit SR step introduces a bulge by a similarity transformation of H with

$$G_1 = \left[\begin{array}{cc|cc} c_1 & & s_1 & \\ & I_{n-1} & & \\ \hline -s_1 & & c_1 & \\ & & & I_{n-1} \end{array}\right]. \tag{5.1}$$

This transformation yields

$$H_1 = G_1 H G_1^{-1} = \left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & & & & & \beta_1' & \zeta_2' & & & \\ b_1 & \delta_2 & & & & \zeta_2' & \beta_2 & \zeta_3 & & \\ & & \delta_3 & & & & \zeta_3 & \ddots & \ddots & \\ & & & \ddots & & & & \ddots & \beta_{n-1} & \zeta_n \\ & & & & \delta_n & & & & \zeta_n & \beta_n \\ \hline \nu_1' & & & & & -\widetilde{\delta}_1 & -b_1 & & & \\ & \nu_2 & & & & & -\delta_2 & & & \\ & & \nu_3 & & & & & -\delta_3 & & \\ & & & \ddots & & & & & \ddots & \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl} \widetilde{\delta}_1 &=& (c_1^2 - s_1^2)\delta_1 + c_1 s_1(\nu_1 + \beta_1), & \zeta_2' &=& c_1\zeta_2, \\ \beta_1' &=& c_1^2\beta_1 - s_1^2\nu_1 - 2c_1 s_1\delta_1, & b_1 &=& s_1\zeta_2, \\ \nu_1' &=& c_1^2\nu_1 - s_1^2\beta_1 - 2c_1 s_1\delta_1. & & & \end{array}$$

Now we will restore the J-Hessenberg form by chasing the bulge b_1 down the diagonal. In order to do so, we will apply the algorithm 'Reduction to J-Hessenberg form' which is given in Table 2.6 to H_1. A symplectic matrix S_2 is constructed such that

$$\widetilde{H} = S_2 H_1 S_2^{-1} \tag{5.2}$$

is in J-Hessenberg form. Due to the special form of H_1, the algorithm simplifies considerably.

First a symplectic Gauss transformation L_2 where the parameters are given by $[c_2, d_2] =$ **gauss1**(b_1, ν_1)

$$L_2 = \left[\begin{array}{ccc|ccc} c_2 & & & & d_2 & \\ & c_2 & & d_2 & & \\ & & I_{n-2} & & & \\ \hline & & & c_2^{-1} & & \\ & & & & c_2^{-1} & \\ & & & & & I_{n-2} \end{array}\right]$$

is applied to eliminate the $(1,2)$ element, resulting in

$$H_2 = L_2 H_1 L_2^{-1} = \left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & b_2 & & & & \widetilde{\beta}_1 & \zeta_2'' & & & \\ & \delta_2 & & & & \zeta_2'' & \beta_2' & \zeta_3' & & \\ & & \delta_3 & & & & \zeta_3' & \ddots & \ddots & \\ & & & \ddots & & & & \ddots & \beta_{n-1} & \zeta_n \\ & & & & \delta_n & & & & \zeta_n & \beta_n \\ \hline \widetilde{\nu}_1 & & & & & -\widetilde{\delta}_1 & & & & \\ & \nu_2' & & & & -b_2 & -\delta_2 & & & \\ & & \nu_3 & & & & & -\delta_3 & & \\ & & & \ddots & & & & & \ddots & \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl}
\widetilde{\beta}_1 &=& -d_2^2\nu_2 + c_2^2\beta_1', & \widetilde{\nu}_1 &=& c_2^{-2}\nu_1', \\
\beta_2' &=& c_2^2\beta_2 - c_2d_2b_1, & \nu_2' &=& c_2^{-2}\nu_2, \\
\zeta_2'' &=& -d_2c_2(\delta_2 + \widetilde{\delta}_1) + c_2^2\zeta_2', & b_2 &=& d_2c_2^{-1}\nu_2, \\
\zeta_3' &=& c_2\zeta_3. &&&
\end{array}$$

The bulge has moved from the $(2,1)$ position to the $(1,2)$ position. Due to the Hamiltonian structure, simultaneously the $(n+1, n+2))$ position moved to the $(n+2, n+1)$ position. Hence, the first column of H_2 is in the desired form. Next the $(n+1)$st column is treated. A symplectic Givens transformation G_2 where the parameters c_3, s_3 are given by $[c_3, s_3] =$ **givens**(ζ_2'', b_2)

$$G_2 = \left[\begin{array}{ccc|ccc}
1 & & & & & \\
& c_3 & & s_3 & & \\
& & I_{n-2} & & & \\
\hline
& & & 1 & & \\
& -s_3 & & & c_3 & \\
& & & & & I_{n-2}
\end{array}\right]$$

is used to eliminate the $(n+2, n+1)$ entry (and simultaneously the $(1,2)$ entry). This gives $H_3 = G_2H_2G_2^T$

$$H_3 = \left[\begin{array}{ccccccc|ccccccc}
\widetilde{\delta}_1 & & & & & & & \widetilde{\beta}_1 & \zeta_2''' & & & & & \\
& \widetilde{\delta}_2 & & & & & & \zeta_2''' & \beta_2'' & \check{\zeta}_3 & & & & \\
& b_3 & \delta_3 & & & & & & \zeta_3'' & \beta_3 & \zeta_4 & & & \\
& & & \delta_4 & & & & & & \zeta_4 & \ddots & \ddots & & \\
& & & & \ddots & & & & & & \ddots & \ddots & \zeta_n & \\
& & & & & & \delta_n & & & & & \zeta_n & \delta_n & \\
\hline
\widetilde{\nu}_1 & & & & & & & -\widetilde{\delta}_1 & & & & & & \\
& \nu_2'' & & & & & & & -\widetilde{\delta}_2 & -b_3 & & & & \\
& & \nu_3 & & & & & & & -\delta_3 & & & & \\
& & & \nu_4 & & & & & & & -\delta_4 & & & \\
& & & & \ddots & & & & & & & \ddots & & \\
& & & & & & \nu_n & & & & & & & -\delta_n
\end{array}\right],$$

where

$$\begin{array}{lcllcl}
\widetilde{\delta}_2 &=& (c_3^2 - s_3^2)\delta_2 - c_3s_3(\nu_2' + \beta_2'), & \zeta_2''' &=& c_3\zeta_2'' - s_3b_2, \\
\beta_2'' &=& c_3^2\beta_2' - 2c_3s_3\delta_2 - s_3^2\nu_2', & \zeta_3'' &=& c_3\zeta_3', \\
\nu_2'' &=& c_3^2\nu_2' - 2c_3s_3\delta_2 - s_3^2\beta_2'', & b_3 &=& s_2\zeta_3'.
\end{array}$$

The first and the $(n+1)$st column of H_3 are in the desired form. The bulge has been chased from position $(2,1)$ $((n+1, n+2))$ to position $(3,2)$ $((n+2, n+3))$. Continuing in the same fashion, it can be chased until the J-Hessenberg form has been restored. In order to derive an algorithm that works only on the parameters which determine the Hamiltonian matrix H, let us consider the next step of the process as well. We will see that $\widetilde{\delta}_1, \widetilde{\delta}_2, \widetilde{\nu}_1$ and $\widetilde{\beta}_1$ will not be changed in the following calculations. They belong to the set of parameters which determine the matrix $\widetilde{H}$ (5.2).

In the next step, first a symplectic Gauss transformation L_3

$$L_3 = \left[\begin{array}{ccccc|ccccc} 1 & & & & & & & & & \\ & c_4 & & & & & & d_4 & & \\ & & c_4 & & & & d_4 & & & \\ & & & I_{n-3} & & & & & & \\ \hline & & & & & 1 & & & & \\ & & & & & & c_4^{-1} & & & \\ & & & & & & & c_4^{-1} & & \\ & & & & & & & & I_{n-3} & \end{array}\right]$$

with $[c_4, d_4] =$ **gauss1**(b_2, ν_2'') to eliminate the $(3,2)$ element is applied, resulting in $H_4 = L_3 H_3 L_3^{-1}$

$$H_4 = \left[\begin{array}{ccccccc|ccccccc} \widetilde{\delta}_1 & & & & & & & \widetilde{\beta}_1 & \widetilde{\zeta}_2 & & & & & \\ & \widetilde{\delta}_2 & b_4 & & & & & \widetilde{\zeta}_2 & \widetilde{\beta}_2 & \zeta_3''' & & & & \\ & & \delta_3 & & & & & & \zeta_3''' & \beta_3' & \zeta_4' & & & \\ & & & \delta_4 & & & & & & \zeta_4' & \beta_4 & \zeta_5 & & \\ & & & & \delta_5 & & & & & & \zeta_5 & \beta_5 & \ddots & \\ & & & & & \ddots & & & & & & \ddots & \ddots & \zeta_n \\ & & & & & & \delta_n & & & & & & \zeta_n & \delta_n \\ \hline \widetilde{\nu}_1 & & & & & & & -\widetilde{\delta}_1 & & & & & & \\ & \widetilde{\nu}_2 & & & & & & & -\widetilde{\delta}_2 & & & & & \\ & & \nu_3' & & & & & & -b_4 & -\delta_3 & & & & \\ & & & \nu_4 & & & & & & & -\delta_4 & & & \\ & & & & \nu_5 & & & & & & & -\delta_5 & & \\ & & & & & \ddots & & & & & & & \ddots & \\ & & & & & & \nu_n & & & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl} \widetilde{\beta}_2 &=& -d_4^2\nu_3 + c_4^2\beta_2'', & \zeta_4' &=& c_4\zeta_4, \\ \beta_3' &=& c_4^2\beta_3 - c_4 d_4 b_3, & \widetilde{\nu}_2 &=& c_4^{-2}\nu_2'', \\ \zeta_2'''' &=& c_4\zeta_2''', & \nu_3' &=& c_4^{-2}\nu_3, \\ \zeta_3''' &=& -d_4 c_4(\delta_3 + \widetilde{\delta}_2) + c_4^2\zeta_3'', & b_4 &=& d_4 c_4^{-1}\nu_3. \end{array}$$

Comparing these computations with those of generating H_2, we find that the set of parameters is transformed in the same way as before, in the formulae just the indices of the parameters have been increased by one. But there is an additional computation updating ζ_2 (as there is no ζ_1 such an update did not occur in the computation of H_2).

Next a symplectic Givens transformation G_3 with $[c_5, s_5] =$ **givens**(ζ_3''', b_4)

$$G_3 = \left[\begin{array}{ccccc|ccccc} 1 & & & & & & & & & \\ & 1 & & & & & & & & \\ & & c_5 & & & & & s_5 & & \\ & & & I_{n-3} & & & & & & \\ \hline & & & & & 1 & & & & \\ & & & & & & 1 & & & \\ & & -s_5 & & & & & c_5 & & \\ & & & & & & & & I_{n-3} & \end{array}\right]$$

is used to eliminate the $(2,3)$ entry. The similarity transformation $H_5 = G_3 H_5 G_3^T$ yields

$$H_5 = \left[\begin{array}{ccccccc|ccccccc}
\widetilde{\delta}_1 & & & & & & & \widetilde{\beta}_1 & \widetilde{\zeta}_2 & & & & & \\
 & \widetilde{\delta}_2 & & & & & & \widetilde{\zeta}_2 & \widetilde{\beta}_2 & \zeta_3'''' & & & & \\
 & & \widetilde{\delta}_3 & & & & & & \zeta_3'''' & \beta_3'' & \zeta_4' & & & \\
 & & b_5 & \delta_4 & & & & & & \zeta_4' & \beta_4 & \zeta_5 & & \\
 & & & & \delta_5 & & & & & & \zeta_5 & \beta_5 & \ddots & \\
 & & & & & \ddots & & & & & & \ddots & \ddots & \zeta_n \\
 & & & & & & \delta_n & & & & & & \zeta_n & \delta_n \\
\hline
\widetilde{\nu}_1 & & & & & & & -\widetilde{\delta}_1 & & & & & & \\
 & \widetilde{\nu}_2 & & & & & & & -\widetilde{\delta}_2 & & & & & \\
 & & \nu_3'' & & & & & & & -\widetilde{\delta}_3 & -b_5 & & & \\
 & & & \nu_4 & & & & & & & -\delta_4 & & & \\
 & & & & \nu_5 & & & & & & & -\delta_5 & & \\
 & & & & & \ddots & & & & & & & \ddots & \\
 & & & & & & \nu_n & & & & & & & -\delta_n
\end{array}\right],$$

where

$$\begin{array}{lcllcl}
\widetilde{\delta}_3 &=& (c_5^2 - s_5^2)\delta_3 - c_5 s_5(\nu_3' + \beta_3'), & \zeta_3'''' &=& c_5\zeta_3''' - s_5 b_4, \\
\beta_3'' &=& c_5^2\beta_3' - 2c_5 s_5\delta_3 - s_5^2\nu_3', & \zeta_4' &=& c_5\zeta_4, \\
\nu_3'' &=& c_5^2\nu_3' - 2c_5 s_5\delta_3 - s_5^2\beta_3', & b_5 &=& s_5\zeta_4.
\end{array}$$

Comparing these computations with those of generating H_3, we find that the set of parameters is transformed in the same way as before, in the formulae just the indices of the parameters have been increased by one. The bulge has been chased down another row and column, we have the same situation as in H_3.

The parameters $\widetilde{\delta}_1, \widetilde{\delta}_2, \widetilde{\delta}_3, \widetilde{\beta}_1, \widetilde{\beta}_2, \widetilde{\zeta}_2, \widetilde{\nu}_1$ and $\widetilde{\nu}_2$ of the resulting matrix $\widetilde{H}$ can be read off. In general, once the bulge is chased down j rows and columns, the leading $j-1$ rows and columns of each block are not changed anymore. The parameters $\widetilde{\delta}_1, \ldots, \widetilde{\delta}_{j+1}, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_j, \widetilde{\zeta}_2, \ldots, \widetilde{\zeta}_{j-1}, \widetilde{\nu}_1, \ldots, \widetilde{\nu}_j$ of the resulting matrix $\widetilde{H}$ can be read off.

From the given reduction it is easy to derive an algorithm that computes the parameters of $\widetilde{H}$ one set (that is, $\widetilde{\delta}_{j+1}, \widetilde{\beta}_j, \widetilde{\zeta}_j, \widetilde{\nu}_j$) at a time given the parameters of H. One has to be careful when the columns $n-1$ and $2n-1$ are treated, as there is no ζ_{n+1}. The non-parameterized version of the algorithm is summarized in Table 5.1. A corresponding MATLAB programme called `param_sr_implicit_single` working only on the parameters is given in Appendix B. A careful flop count reveals

- the computation of the parameters of a Givens transformation requires 4 multiplications, 1 addition and 1 square root,
- the computation of the parameters of a Gauss transformation requires 4 multiplications, 1 addition and 2 square roots,
- the introduction of the bulge requires 15 multiplications and 7 additions,
- the application of the Gauss transformation requires in general 15 multiplications and 4 additions (in the last step, only 14 multiplications are needed),
- the application of the Givens transformation requires in general 16 multiplications and 11 additions (in the last step, only 13 multiplications are needed),

Algorithm: Single shift implicit SR step

Given a $2n \times 2n$ Hamiltonian matrix A in J-Hessenberg form compute a single shift implicit SR step. That is, given a real shift μ, the symplectic matrix S of the SR decomposition $A - \mu I = SR$ is computed implicitly. A will be overwritten by its new J–Hessenberg form SAS^{-1}.

compute G_1 such that $G_1((\delta_1 - \mu)e_1 + \nu_1 e_{n+1}) = \alpha e_1$
$A = G_1 A G_1^T$
$S = G_1$
for $j = 1 : n-1$
  if $A_{j+1,j} \neq 0$ and $A_{n+j,n+j} = 0$
  then stop, reduction does not exist
  end
  compute L_{j+1} such that $(L_{j+1}A)_{j+1,j} = 0$
  $A = L_{j+1} A L_{j+1}^{-1}$
  $S = L_{j+1} S$
  compute G_{j+1} such that $(G_{j+1}A)_{n+j+1,n+j} = 0$
  $A = G_{j+1} A G_{j+1}^T$
  $S = G_{j+1} S$
end

TABLE 5.1
Single implicit SR step

so that the first step of the algorithm (Givens transformation to introduce the bulge) requires 19 multiplications, 8 additions and 1 square root, followed by repeated applications of a Gauss and a Givens transformation which requires $(n-1)\times$ 39 multiplications, 17 additions and 3 square roots (minus 2 multiplications in the last step). Hence, the algorithm requires $39n - 22$ multiplications, $17n - 9$ additions and $3n - 2$ square roots. In other words, the implicit single SR step working on the parameters only requires $\mathcal{O}(n)$ flops. As the entire process works only on the parameters which determine the Hamiltonian matrix, the Hamiltonian structure is forced in every step of the algorithm.

5.2. A double shift implicit SR step. Now let us consider a double shift implicit SR step. As H is a Hamiltonian matrix, for a double shift the shift polynomial $q_2(H) = (H - \mu I)(H + \mu I)$ should be chosen for $\mu \in \mathbb{R}$ or $\mu = i\omega, \omega \in \mathbb{R}$. In the first case, the first column of q_2 is of the form

$$x = q_2(H)e_1 = (H^2 - \mu^2 I)e_1 = (\delta_1^2 + \nu_1\beta_1 - \mu^2)e_1 + \nu_1\zeta_2 e_2.$$

while in the second case the first column of q_2 is of the form

$$x = q_2(H)e_1 = (H^2 + \omega^2 I)e_1 = (\delta_1^2 + \nu_1\beta_1 + \omega^2)e_1 + \nu_1\zeta_2 e_2.$$

In case, a Rayleigh-quotient strategy is used, the shifts will be chosen as the eigenvalues of

$$H_{2\times 2} = \left[\begin{array}{c|c} \delta_n & \beta_n \\ \hline \nu_n & -\delta_n \end{array}\right].$$

There is no need to compute the eigenvalues of $H_{2\times 2}$ directly. Comparing $q_2(H) = (H-\mu I)(H+\mu I) = H^2 - \mu^2 I$ with the characteristic polynomial of $H_{2\times 2}$

$$\det(H_{2\times 2} - \lambda I) = \lambda^2 - (\delta_n^2 + \beta_n\nu_n)$$

yields that q_2 should be chosen as

$$q_2(H) = H^2 - (\delta_n^2 + \beta_n\nu_n)I.$$

The first column is then given by

$$q_2(H)e_1 = (\delta_1^2 + \nu_1\beta_1 - \delta_n^2 - \beta_n\nu_n)e_1 + \nu_1\zeta_2 e_2.$$

Hence, in any case, the first column of q_2 has the same form. This vector can be transformed into a multiply of e_1 by a symplectic Givens transformation of type II

$$\widetilde{G}_1 = \left[\begin{array}{cc|cc} c_1 & s_1 & & \\ -s_1 & c_1 & & \\ & & I_{n-2} & \\ \hline & & c_1 & s_1 \\ & & -s_1 & c_1 \\ & & & & I_{n-2} \end{array}\right]. \tag{5.3}$$

with $[c_1, s_1] = \mathbf{givens}(\delta_1^2 + \nu_1\beta_1 - s^2, \nu_1\zeta_2)$ where $s = \mu$, $s = i\omega$ or $s^2 = \delta_n^2 + \beta_n\nu_n$ depending on the choice of the shift. Hence, the bulge is introduced by a similarity transformation of H with $\widetilde{G}_1$. This transformation yields

$$H_1 = \widetilde{G}_1 H \widetilde{G}_1^T = \left[\begin{array}{cccccc|cccccc} \widetilde{\delta}_1 & b_1 & & & & & \beta_1' & \zeta_2' & b_3 & & & \\ b_1 & \delta_2' & & & & & \zeta_2' & \beta_2' & \zeta_3' & & & \\ & & \delta_3 & & & & b_3 & \zeta_3' & \beta_3 & \zeta_4 & & \\ & & & \delta_4 & & & & & \zeta_4 & \ddots & \ddots & \\ & & & & \ddots & & & & & \ddots & \ddots & \zeta_n \\ & & & & & \delta_n & & & & & \zeta_n & \beta_n \\ \hline \nu_1' & b_2 & & & & & -\widetilde{\delta}_1 & -b_1 & & & & \\ b_2 & \nu_2' & & & & & -b_1 & -\delta_2' & & & & \\ & & \nu_3 & & & & & & -\delta_3 & & & \\ & & & \nu_4 & & & & & & -\delta_4 & & \\ & & & & \ddots & & & & & & \ddots & \\ & & & & & \nu_n & & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{aligned} \widetilde{\delta}_1 &= c_1^2\delta_1 + s_1^2\delta_2, & \zeta_3' &= c_1\zeta_3, \\ \delta_2' &= c_1^2\delta_2 + s_1^2\delta_1, & b_1 &= c_1 s_1(\delta_2 - \delta_1), \\ \nu_1' &= c_1^2\nu_1 + s_1^2\nu_2, & b_2 &= c_1 s_1(\nu_2 - \nu_1), \\ \nu_2' &= c_1^2\nu_2 + s_1^2\nu_1, & b_3 &= s_1\zeta_3, \\ \beta_1' &= c_1^2\beta_1 + s_1^2\beta_2 + 2c_1 s_1\zeta_2, & & \\ \beta_2' &= s_1^2\beta_1 + c_1^2\beta_2 - 2c_1 s_1\zeta_2, & & \\ \zeta_2' &= (c_1^2 - s_1^2)\zeta_2 + c_1 s_1(\beta_2 - \beta_1). & & \end{aligned}$$

The $(1,1)$ block (and hence, consequently the $(2,2)$ block) is symmetric although this is not forced by the Hamiltonian structure. This property will be destroyed in

the next step, but it will be retained later. Now we will restore the J-Hessenberg form by applying Algorithm 2.6, that is a symplectic matrix S_2 is constructed such that

$$\widetilde{H} = S_2 H_1 S_2^{-1} \tag{5.4}$$

is in J-Hessenberg form. As in the previous section, the algorithm simplifies due to the special structure of H_1. First a symplectic Givens transformation G_2

$$G_2 = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ & c_2 & & & s_2 & \\ & & I_{n-2} & & & \\ \hline & & & 1 & & \\ & -s_2 & & & c_2 & \\ & & & & & I_{n-2} \end{array}\right]$$

with $[c_2, s_2] =$ **givens**(b_1, b_2) is used to eliminate the bulge element b_2 in the $(n+2, 1)$ position. This gives $H_2 = G_2 H_1 G_2^T$

$$H_2 = \left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & b_2' & & & & \beta_1' & \zeta_2'' & b_3 & & \\ b_1' & \delta_2'' & & & & \zeta_2'' & \beta_2'' & \zeta_3'' & & \\ & b_4 & \delta_3 & & & b_3 & \zeta_3'' & \beta_3 & \zeta_4 & \\ & & & \delta_4 & & & & \zeta_4 & \ddots & \ddots \\ & & & & \ddots & & & & \ddots & \ddots \\ & & & & \delta_n & & & & & \zeta_n \\ & & & & & & & & \zeta_n & \beta_n \\ \hline \nu_1' & & & & & -\widetilde{\delta}_1 & -b_1' & & & \\ & \nu_2'' & & & & -b_2' & -\delta_2'' & -b_4 & & \\ & & \nu_3 & & & & & -\delta_3 & & \\ & & & \nu_4 & & & & & -\delta_4 & \\ & & & & \ddots & & & & & \ddots \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl} \delta_2'' &=& (c_2^2 - s_2^2)\delta_2' + c_2 s_2(\nu_2' + \beta_2'), & \zeta_3'' &=& c_2\zeta_3', \\ \beta_2'' &=& -s_2^2\nu_2' - 2c_2 s_2 \delta_2' + c_2^2\beta_2', & b_1' &=& c_2 b_1 + s_2 b_2, \\ \nu_2'' &=& c_2^2\nu_2' - 2c_2 s_2\delta_2' - s_2^2\beta_2', & b_2' &=& c_2 b_1 + s_2\zeta_2', \\ \zeta_2'' &=& c_2\zeta_2' - s_2 b_1, & b_4 &=& s_2\zeta_3'. \end{array}$$

The $(1,1)$ block is no longer symmetric, an additional bulge element is created by this transformation.

Next a symplectic Gauss transformation

$$L_1 = \left[\begin{array}{ccc|ccc} c_3 & & & & d_3 & \\ & c_3 & & d_3 & & \\ & & I_{n-2} & & & \\ \hline & & & c_3^{-1} & & \\ & & & & c_3^{-1} & \\ & & & & & I_{n-2} \end{array}\right]$$

with $[c_3, d_3] =$ **gauss1**(b'_1, ν'_1) is used to eliminate the element in position $(2,1)$. The transformation $H_3 = L_1 H_2 L_1^{-1}$ does not create any new entry;

$$H_3 = \left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & b''_2 & & & & \beta''_1 & \zeta'''_2 & b'_3 & & \\ & \delta''_2 & & & & \zeta'''_2 & \beta'''_2 & \zeta'''_3 & & \\ & b'_4 & \delta_3 & & & b'_3 & \zeta'''_3 & \beta_3 & \zeta_4 & \\ & & & \delta_4 & & & & \zeta_4 & \ddots & \ddots \\ & & & & \ddots & & & & \ddots & \ddots \\ & & & & & & & & & \zeta_n \\ & & & & \delta_n & & & & \zeta_n & \beta_n \\ \hline \nu''_1 & & & & & -\widetilde{\delta}_1 & & & & \\ & \nu'''_2 & & & & -b''_2 & -\delta''_2 & -b'_4 & & \\ & & \nu_3 & & & & & -\delta_3 & & \\ & & & \nu_4 & & & & & -\delta_4 & \\ & & & & \ddots & & & & & \ddots \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl} \beta''_1 &=& c_3^2\beta'_1 - 2c_3 d_3 b'_2 - d_3^2\nu''_2, & \nu'''_2 &=& c_3^{-2}\nu''_2, \\ \beta'''_2 &=& c_3^2\beta''_2 - c_3 d_3 b'_1, & b''_2 &=& b'_2 + d_3\nu''_2/c_3, \\ \zeta'''_2 &=& c_3^2\zeta''_2 - c_3 d_3(\widetilde{\delta}_1 + \delta''_2), & b'_3 &=& c_3 b_3 - d_3 b_4, \\ \zeta'''_3 &=& c_3\zeta''_3, & b'_4 &=& b_4/c_3, \\ \nu''_1 &=& c_3^{-2}\nu'_1. & & & \end{array}$$

The first column of H_3 is in the desired form. Next column $n+1$ is treated. But let us note first that

$$\operatorname{rank}\left(\begin{bmatrix} b''_2 & \zeta'''_2 \\ b'_4 & \zeta'''_3 \end{bmatrix}\right) = 1. \tag{5.5}$$

In order to see this we have to have a closer look at the entries of this 2×2 matrix:

$$\begin{aligned} b'_4 &= b_4/c_3 = s_2\zeta'_3/c_3 = s_2 c_1 \zeta_3/c_3, \\ \zeta'''_3 &= c_3\zeta''_3 = c_3 c_2 \zeta'_3 = c_3 c_2 c_1 \zeta_3. \end{aligned}$$

Hence,

$$\zeta'''_3 = c_3^2 c_2 b'_4/s_2.$$

So, if we can show that b''_2 and ζ'''_2 satisfy the same relation

$$\zeta'''_2 = c_3^2 c_2 b''_2/s_2, \tag{5.6}$$

then the above 2×2 matrix has to have rank 1. See Appendix A for a detailed derivation of this relation.

Next a symplectic Givens transformation

$$G'_2 = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ & c_4 & & & s_4 & \\ & & I_{n-2} & & & \\ \hline & & & 1 & & \\ & -s_4 & & & c_4 & \\ & & & & & I_{n-2} \end{array}\right]$$

with $[c_4, s_4] =$ **givens**$(\zeta_2'''', -b_2'')$ is applied to eliminate the bulge element b_2''. This gives $H_4 = G_2' H_3 (G_2')^T$

$$H_4 = \left[\begin{array}{ccccccc|ccccccc}
\widetilde{\delta}_1 & & & & & & & \beta_1'' & \zeta_2'''' & b_3' & & & & \\
 & \delta_2'''' & & & & & & \zeta_2'''' & \beta_2'''' & \zeta_3'''' & & & & \\
 & & \delta_3 & & & & & b_3' & \zeta_3'''' & \beta_3 & \zeta_4 & & & \\
 & & & \delta_4 & & & & & & \zeta_4 & \ddots & \ddots & & \\
 & & & & \ddots & & & & & & \ddots & \ddots & \zeta_n & \\
 & & & & & \delta_n & & & & & & \zeta_n & \beta_n & \\
\hline
\nu_1'' & & & & & & & -\widetilde{\delta}_1 & & & & & & \\
 & \nu_2'''' & & & & & & & -\delta_2'''' & & & & & \\
 & & \nu_3 & & & & & & & -\delta_3 & & & & \\
 & & & \nu_4 & & & & & & & -\delta_4 & & & \\
 & & & & \ddots & & & & & & & \ddots & & \\
 & & & & & \nu_n & & & & & & & -\delta_n &
\end{array}\right],$$

where

$$\begin{array}{rclcrcl}
\delta_2'''' &=& c_4^2\delta_2'' + c_4 s_4(\nu_2'''' + \beta_2'''') - s_4^2\delta_2'', & & \zeta_2'''' &=& -s_4 b_2'' + c_4\zeta_2'''', \\
\beta_2'''' &=& c_4^2\beta_2'''' - 2c_4 s_4\delta_2'' - s_4^2\nu_2'''', & & \zeta_3'''' &=& -s_4 b_4' + c_4\zeta_3'''', \\
\nu_2'''' &=& c_4^2\nu_2'''' - 2c_4 s_4\delta_2'' - s_4^2\beta_2''''. & & & & \\
\end{array}$$

No new entry is created. In particular, we have

$$\begin{bmatrix} c_4 & s_4 \\ -s_4 & c_4 \end{bmatrix} \begin{bmatrix} b_2'' & \zeta_2'''' \\ b_4' & \zeta_3'''' \end{bmatrix} \begin{bmatrix} c_4 & -s_4 \\ s_4 & c_4 \end{bmatrix} = \begin{bmatrix} 0 & \zeta_2'''' \\ b_4'' & \zeta_3'''' \end{bmatrix},$$

due to the way c_4 and s_4 are chosen. But as the rank of the modified 2×2 matrix is 1, it follows that

$$b_4'' = 0.$$

Next the bulge element b_3' is annihilated by a symplectic Givens transformation of type II

$$\widetilde{G}_2 = \left[\begin{array}{ccccc|ccccc}
1 & & & & & & & & & \\
 & c_5 & s_5 & & & & & & & \\
 & -s_5 & c_5 & & & & & & & \\
 & & & I_{n-3} & & & & & & \\
\hline
 & & & & & 1 & & & & \\
 & & & & & & c_5 & s_5 & & \\
 & & & & & & -s_5 & c_5 & & \\
 & & & & & & & & I_{n-3} &
\end{array}\right]$$

with $[c_5, s_5] = \mathbf{givens}(\zeta_2'''', b_3')$. This yields

$$H_5 = \left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & & & & & \widetilde{\beta}_1 & \zeta_2'''''' & & & \\ & \widetilde{\delta}_2 & b_1 & & & \zeta_2'''''' & \beta_2'''''' & \zeta_3'''''' & b_3 & \\ & b_1 & \delta_3' & & & & \zeta_3'''''' & \beta_3' & \zeta_4' & \\ & & & \delta_4 & & & b_3 & \zeta_4' & \ddots & \ddots \\ & & & & \ddots & & & & \ddots & \ddots \\ & & & & \delta_n & & & & & \zeta_n \\ & & & & & & & & \zeta_n & \beta_n \\ \hline \widetilde{\nu}_1 & & & & & -\widetilde{\delta}_1 & & & & \\ & \nu_2'''''' & b_2 & & & & -\widetilde{\delta}_2 & -b_1 & & \\ & b_2 & \nu_3' & & & & -b_1 & -\delta_3' & & \\ & & & \nu_4 & & & & & -\delta_4 & \\ & & & & \ddots & & & & & \ddots \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right],$$

where

$$\begin{array}{lcllcl} \delta_2'''' & = & c_5^2\delta_2''' + s_5^2\delta_3, & \zeta_4' & = & c_5\zeta_4, \\ \delta_3' & = & c_5^2\delta_3 + s_5^2\delta_2''', & \nu_2'''''' & = & c_5^2\nu_2'''' + s_5^2\nu_3, \\ \beta_2'''''' & = & c_5^2\beta_2'''' + 2c_5s_5\zeta_3'''' + s_5^2\beta_3, & \nu_3' & = & c_5^2\nu_3 + s_5^2\nu_2'''', \\ \beta_3' & = & c_5^2\beta_3 - 2c_5s_5\zeta_3'''' + s_5^2\beta_2'''', & b_1 & = & c_5s_5(\delta_3 - \delta_2''''), \\ \zeta_2'''''' & = & c_5\zeta_2'''' + s_5b_3', & b_2 & = & c_5s_5(\nu_3 - \nu_2''''), \\ \zeta_3'''''' & = & c_5^2\zeta_3'''' + c_5s_5(\beta_3 - \beta_2'''') - s_5^2\zeta_3'''', & b_3 & = & s_5\zeta_4. \end{array}$$

The first and the $(n+1)$st column of H_5 are in the desired form. The situation is the same as in H_1, just the bulge has moved one row and one column further down the diagonal in each block. Continuing in the same fashion, it can be chased until the J-Hessenberg form has been restored. In order to derive an algorithm that works only on the parameters which determine the Hamiltonian matrix H, let us consider the next step of the process as well. We will see that $\widetilde{\delta}_1, \widetilde{\delta}_2 = \delta_2'''', \widetilde{\nu}_1 = \nu_1''$ and $\widetilde{\beta}_1 = \beta_1''$ will not be changed in the following calculations. They belong to the set of parameters which will determine the matrix $\widetilde{H}$ (5.4).

As in H_1, the $(1,1)$ block (and hence, consequently the $(2,2)$ block) is symmetric although this is not forced by the Hamiltonian structure.

For the next step of the algorithm the second column of H_5 is treated. First a symplectic Givens transformation G_3

$$G_3 = \left[\begin{array}{ccc|ccc} I_2 & & & & & \\ & c_6 & & & s_6 & \\ & & I_{n-3} & & & \\ \hline & & & I_2 & & \\ & -s_6 & & & c_6 & \\ & & & & & I_{n-3} \end{array}\right]$$

with $[c_6, s_6] = \mathbf{givens}(b_1, b_2)$ is used to eliminate the bulge element b_2 in the $(n+3, 2)$

position. This gives $H_6 = G_3 H_5 G_3^T$

$$H_6 = \left[\begin{array}{cccccc|cccccc}
\widetilde{\delta}_1 & & & & & & \widetilde{\beta}_1 & \zeta_2'''''' & & & & \\
 & \widetilde{\delta}_2 & b_2' & & & & \zeta_2'''''' & \beta_2'''''' & \zeta_3'''''' & b_3 & & \\
 & b_1' & \delta_3'' & & & & & \zeta_3'''''' & \beta_3'' & \zeta_4'' & & \\
 & & b_4 & \delta_4 & & & & b_3 & \zeta_4'' & \ddots & \ddots & \\
 & & & & \ddots & & & & & \ddots & \ddots & \zeta_n \\
 & & & & & \delta_n & & & & & \zeta_n & \beta_n \\
\hline
\widetilde{\nu}_1 & & & & & & -\widetilde{\delta}_1 & & & & & \\
 & \nu_2'''''' & & & & & & -\widetilde{\delta}_2 & -b_1' & & & \\
 & & \nu_3'' & & & & & -b_2' & -\delta_3'' & -b_4 & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \ddots & & & & & & \ddots & \\
 & & & & & \nu_n & & & & & & -\delta_n
\end{array}\right],$$

where

$$\begin{aligned}
\delta_3'' &= (c_6^2 - s_6^2)\delta_3' + c_6 s_6(\nu_3' + \beta_3'), \\
\beta_3'' &= -s_6^2\nu_3' - 2c_6 s_6 \delta_3' + c_6^2 \beta_3', \\
\zeta_3'''''' &= c_6 \zeta_3'''''' - s_6 b_1, \\
\zeta_4'' &= c_6 \zeta_4', \\
\nu_3'' &= c_6^2 \nu_3' - 2c_6 s_6 \delta_3' - s_6^2 \beta_3', \\
b_1' &= c_6 b_1 + s_6 b_2, \\
b_2' &= c_2 b_1 + s_2 \zeta_3'''''', \\
b_4 &= s_6 \zeta_4'.
\end{aligned}$$

Comparing these computations with those of H_2, we find that the next set of parameters is transformed in the same way as before, in the formulae just the indices have been increased by one. As before, the $(1,1)$ block is no longer symmetric, an additional bulge element is created by this transformation.

Next a symplectic Gauss transformation

$$L_2 = \left[\begin{array}{ccccc|ccccc}
1 & & & & & & & & & \\
 & c_7 & & & & & & & d_7 & \\
 & & c_7 & & & & & d_7 & & \\
 & & & I_{n-3} & & & & & & \\
\hline
 & & & & & 1 & & & & \\
 & & & & & & c_7^{-1} & & & \\
 & & & & & & & c_7^{-1} & & \\
 & & & & & & & & I_{n-3} &
\end{array}\right]$$

with $[c_7, d_7] =$ **gauss1**(b_1', ν_2'''''') is used to eliminate the element in position $(3,2)$.

The transformation $H_7 = L_2 H_6 L_2^{-1}$ does not create any new entry.

$$H_7 = \left[\begin{array}{cccccc|cccccc}
\widetilde{\delta}_1 & & & & & & \widetilde{\beta}_1 & \zeta_2'''''' & & & & \\
 & \widetilde{\delta}_2 & b_2'' & & & & \zeta_2'''''' & \beta_2'''''' & \zeta_3'''''''' & b_3' & & \\
 & & \delta_3'' & & & & & \zeta_3'''''' & \beta_3''' & \zeta_4''' & & \\
 & & b_4' & \delta_4 & & & & b_3' & \zeta_4''' & \ddots & \ddots & \\
 & & & & \ddots & & & & & \ddots & \ddots & \zeta_n \\
 & & & & & \delta_n & & & & & \zeta_n & \beta_n \\
\hline
\widetilde{\nu}_1 & & & & & & -\widetilde{\delta}_1 & & & & & \\
 & \nu_2'''''' & & & & & & -\widetilde{\delta}_2 & & & & \\
 & & \nu_3''' & & & & & -b_2'' & -\delta_3'' & -b_4' & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \ddots & & & & & & \ddots & \\
 & & & & & \nu_n & & & & & & -\delta_n
\end{array}\right],$$

where

$$\begin{array}{lcllcl}
\beta_2'''''' &=& c_7^2\beta_2'''''' - 2c_7d_7b_2' - d_7^2\nu_3'', & \nu_3''' &=& c_7^{-2}\nu_3'', \\
\beta_3''' &=& c_7^2\beta_3'' - c_7d_7b_1', & b_2'' &=& b_2' + d_7\nu_3''/c_7, \\
\zeta_3'''''''' &=& c_7^2\zeta_3'''''' - c_7d_7(\widetilde{\delta}_2 + \delta_3''), & b_3' &=& c_7b_3 - d_7b_4, \\
\zeta_4''' &=& c_7\zeta_4'', & b_4' &=& b_4/c_7 \\
\nu_2'''''' &=& c_7^{-2}\nu_2'''''', & \zeta_2'''''' &=& c_7\zeta_2''''''.
\end{array}$$

Comparing these computations with those of generating H_3, we find that the next set of parameters is transformed in the same way as before, in the formulae just the indices of the parameters have been increased by one. But there is an additional computation updating ζ_2 (as there is no ζ_1 such an update did not occur in the computation of H_3).

Let us note first that in complete analogy to (5.5) we have

$$\operatorname{rank}\left(\begin{bmatrix} b_2'' & \zeta_3'''''''' \\ b_4' & \zeta_4''' \end{bmatrix}\right) = 1.$$

Hence, the next two steps which consist of applying a symplectic Givens transformation G_3' to eliminate the entry in position $(3,2)$ and a symplectic Givens transformation $\widetilde{G}_3$ of type II to eliminate the entry in position $(3, n+2)$ are completely analogous to the derivation of H_4 and H_5, in the formulae the indices of the parameters have to be increased by one.

From the given reduction it is easy to derive an algorithm that computes the parameters of $\widetilde{H}$ one set (that is, $\widetilde{\delta}_{j+1}, \widetilde{\beta}_j, \widetilde{\zeta}_j, \widetilde{\nu}_j$) at a time given the parameters of H. One has to be careful when the columns $n-1$ and $2n-1$ are treated, as there is no

ζ_{n+1} and the bulge will be chased out of the matrix. Assume that we have achieved

$$\left[\begin{array}{cccc|cccc}
\ddots & & & & \ddots & & & \\
 & x & & & & x & x & \oplus \\
 & & x & & & x & x & x \\
 & & & x & & \oplus & x & x \\
\hline
\ddots & & & & \ddots & & & \\
 & x & & & & x & & \\
 & & x & & & & x & \\
 & & & x & & & & x
\end{array}\right],$$

where the bulge entries are denotes by $\oplus$ and all other matrix entries by x. This is a situation similar to the one in H_4. Next, a symplectic Givens transformation of type II is used to eliminate the entry in $(n, 2n-3)$. This yields

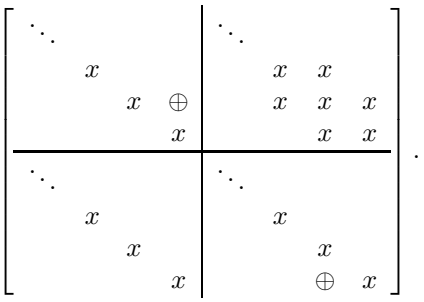
$$\left[\begin{array}{cccc|cccc}
\ddots & & & & \ddots & & & \\
 & x & & & & x & x & \\
 & & x & \oplus & & x & x & x \\
 & & \oplus & x & & & x & x \\
\hline
\ddots & & & & \ddots & & & \\
 & x & & & & x & & \\
 & & x & \oplus & & & x & \oplus \\
 & & \oplus & x & & & \oplus & x
\end{array}\right].$$

One part of the bulge (denoted b_3 in H_5) has been moved out of the matrix, hence the computations needed here do not involve the computation of b_3 (and ζ_{n+1} as this parameter does not exist. For the rest of the discussion in this section it will be assumed that only parameters within the allowed index range are computed.). Next, a symplectic Givens transformation is applied which eliminates the entry $(2n, n-1)$

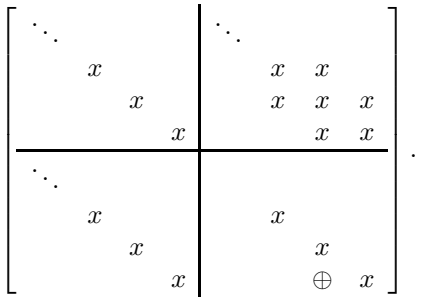
$$\left[\begin{array}{cccc|cccc}
\ddots & & & & \ddots & & & \\
 & x & & & & x & x & \\
 & & x & \oplus & & x & x & x \\
 & & \oplus & x & & & x & x \\
\hline
\ddots & & & & \ddots & & & \\
 & x & & & & x & & \\
 & & x & & & & x & \oplus \\
 & & & x & & & \oplus & x
\end{array}\right].$$

Comparing the necessary computations with those in generating H_2 or H_6, it can be found that the bulge entry b_4 is not generated here. A symplectic Gauss transforma-

tion annihilates the $(n, n-1)$ entry



Neither the bulge entry b'_3 nor the entry b'_4 are generated (compare with the situation in generating H_3 or H_7). A symplectic Givens transformation to eliminate the last bulge entry completes the bulge chase



The symplectic Givens transformation of type II, which was necessary in treating the previous columns, is not needed here. The non-parameterized version of the algorithm is summarized in Table 5.2. A corresponding MATLAB programme called `param_sr_implicit_double` working only on the parameters is given in Appendix B. A careful flop count reveals

- the computation of the parameters of a Givens transformation requires 4 multiplications, 1 addition and 1 square root,
- the computation of the parameters of a Gauss transformation requires 4 multiplications, 1 addition and 2 square roots,
- the introduction of the bulge requires 28 multiplications and 16 additions,
- in the SR step a sequence of Givens, Gauss, Givens and Givens type II is applied where the application of the first Givens transformation requires in general 20 multiplications and 10 additions,
- the application of the Gauss transformation requires in general 20 multiplications and 7 additions,
- the application of the second Givens transformation requires in general 17 multiplications and 9 additions,
- the application of the Givens transformation of type II requires in general 27 multiplications and 14 additions,

so that the first step of the algorithm (Givens transformation to introduce the bulge) requires 32 multiplications, 17 additions and 1 square root, followed by repeated applications of a Givens, a Gauss, a Givens and a Givens type II transformation which requires $(n-1)\times$ 69 multiplications, 29 additions and 4 square roots plus $(n-2)\times$ 31 multiplications, 15 additions and 1 square root.

Algorithm: Double shift implicit SR step

Given a $2n \times 2n$ Hamiltonian matrix A in J-Hessenberg form compute a double shift implicit SR step. That is, given a real shift μ or a purely imaginary shift $\mu = i\omega$, the symplectic matrix S of the SR decomposition $(A - \mu I)(A + \mu I)$ is computed implicitly. A will be overwritten by its new J–Hessenberg form SAS^{-1}.

compute $\widetilde{G}_1$ such that $\widetilde{G}_1((\delta_1^2 + \nu_1\beta_1\mu^2)e_1 + \nu_1\zeta_2 e_2) = \alpha e_1$
$A = \widetilde{G}_1 A \widetilde{G}_1^T$
$S = \widetilde{G}_1$
for $j = 1 : n-1$
  compute G_{j+1} such that $(G_{j+1}A)_{n+j+1,j} = 0$
  $A = G_{j+1} A G_{j+1}^T$
  $S = G_{j+1} S$
  if $A_{j+1,j} \neq 0$ and $A_{n+j,n+j} = 0$
  then stop, reduction does not exist
  end
  compute L_{j+1} such that $(L_{j+1}A)_{j+1,j} = 0$
  $A = L_{j+1} A L_{j+1}^{-1}$
  $S = L_{j+1} S$
  compute G_{j+1} such that $(G_{j+1}A)_{n+j+1,n+j} = 0$
  $A = G_{j+1} A G_{j+1}^T$
  $S = G_{j+1} S$
  if $j < n-1$
  then compute $\widetilde{G}_{j+1}$ such that $(\widetilde{G}_{j+1}A)_{n+j+2,n+j} = 0$
    $A = \widetilde{G}_{j+1} A \widetilde{G}_{j+1}^T$
    $S = \widetilde{G}_{j+1} S$
  end
end

TABLE 5.2
Double shift implicit SR step

Hence, the algorithm requires $100n - 131$ multiplications, $44n - 59$ additions and $5n - 3$ square roots. In other words, the implicit double SR step working on the parameters only requires $\mathcal{O}(n)$ flops. As the entire process works only on the parameters which determine the Hamiltonian matrix, the Hamiltonian structure is forced in every step of the algorithm.

5.3. A quadruple shift implicit SR step. Finally, let us consider an implicitly shifted quadruple shift step. The shift $\gamma \in \mathbb{C}$, $\mathrm{Re}(\gamma) \neq 0$ defines the spectral transformation

$$(H - \gamma I)(H + \gamma I)(H - \overline{\gamma} I)(H + \overline{\gamma} I).$$

An implicitly shifted quadruple step can also be used to perform two double shift steps, in that case we have shifts μ and η that are either real or purely imaginary. The spectral transformation is given by

$$X = (H - \mu I)(H + \mu I)(H - \eta I)(H + \eta I). \tag{5.7}$$

With $\mu = \gamma$ and $\eta = \overline{\gamma}$ this is just the spectral transformation given above. Hence, let us consider the slightly more general expression in our derivations.

First we need to compute the first column of X (5.7) in order to determine the first symplectic transformation which will introduce the bulge to be chased;

$$\begin{aligned} x &= (H-\mu I)(H+\mu I)(H-\eta I)(H+\eta I)e_1 \\ &= (H^2-\mu^2 I)(H^2-\eta^2 I)e_1 \\ &= (H^2-\mu^2 I)[(\delta_1^2+\nu_1\beta_1-\eta^2)e_1+\nu_1\zeta_2 e_2] \\ &= (H^2-\mu^2 I)[(a_1-\eta^2)e_1+\nu_1\zeta_2 e_2] \end{aligned}$$

where we used for notational convenience

$$a_j = \delta_j^2 + \nu_j\beta_j.$$

Hence,

$$\begin{aligned} x \;&=\; (a_1-\eta^2)[(a_1-\mu^2)e_1+\nu_1\zeta_2 e_2] + \nu_1\zeta_2[H(\delta_2 e_2+\nu_2 e_{n+2}-\mu^2 e_2] \\ &= (a_1-\eta^2)[(a_1-\mu^2)e_1+\nu_1\zeta_2 e_2] \\ &\qquad + \nu_1\zeta_2[\delta_2(\delta_2 e_2+\nu_2 e_{n+2}) + \nu_2(\zeta_2 e_1+\beta_2 e_2+\zeta_3 e_3-\delta_2 e_{n+2}) - \mu^2 e_2] \\ &= [(a_1-\eta^2)(a_1-\mu^2)+\nu_1\nu_2\zeta_2^2]e_1 + \nu_1\zeta_2[(a_1+a_2-\eta^2-\mu^2)]e_2 + \nu_1\nu_2\zeta_2\zeta_3 e_3 \\ &= [a_1^2+\nu_1\nu_2\zeta_2^2-(\eta^2+\mu^2)a_1+\mu^2\eta^2]e_1 + \nu_1\zeta_2[(a_1+a_2)-(\eta^2+\mu^2)]e_2 \\ &\qquad + \nu_1\nu_2\zeta_2\zeta_3 e_3 \\ &=: x_1e_1 + x_2e_2 + x_3e_3. \end{aligned}$$

In case, a Raleigh-quotient like shift strategy is chosen, the spectral transformation function q_4 has to be chosen as derived in (4.3)

$$\begin{aligned} q_4(H)e_1 = (a_1^2+\zeta_2^2\nu_1\nu_2 - (a_{n-1}+a_n)a_1 + a_{n-1}a_n - \nu_{n-1}\nu_n\zeta_n^2)e_1 \\ + \zeta_2\nu_1[(a_1+a_2)-(a_{n-1}+a_n)]e_2 + \nu_1\nu_2\zeta_2\zeta_3 e_3. \end{aligned}$$

A symplectic Householder transformation or the product of two symplectic Givens transformations $\widetilde{G}_1^{(1)}\widetilde{G}_2^{(1)}$ can be used to transform this vector into a multiple of e_1:

$$\widetilde{G}_1^{(1)}\widetilde{G}_2^{(1)}x = \alpha e_1$$

for some α where $\widetilde{G}_1^{(1)}\widetilde{G}_2^{(1)}$ is given by

$$\left[\begin{array}{ccc|ccc} c_2 & s_2 & & & & \\ -s_2 & c_2 & & & & \\ & & I_{n-2} & & & \\ \hline & & & c_2 & s_2 & \\ & & & -s_2 & c_2 & \\ & & & & & I_{n-2} \end{array}\right] \left[\begin{array}{ccc|ccc} 1 & & & & & \\ & c_1 & s_1 & & & \\ & -s_1 & c_1 & & & \\ & & I_{n-3} & & & \\ \hline & & & 1 & & \\ & & & & c_1 & s_1 \\ & & & & -s_1 & c_1 \\ & & & & & I_{n-3} \end{array}\right],$$

where $[c_1, s_1] = \mathbf{givens}(x_2, x_3)$ and $[c_2, s_2] = \mathbf{givens}(x_1, c_1x_2 + s_1x_3)$. Applying $\widetilde{G}_2^{(1)}$

yields

$$H_1 = \widetilde{G}_2^{(1)} H (\widetilde{G}_2^{(1)})^T = \left[\begin{array}{ccccc|ccccc} \delta_1 & & & & & \beta_1 & \zeta_2^{(1)} & e_3 & & \\ & \delta_2^{(1)} & e_1 & & & \zeta_2^{(1)} & \beta_2^{(1)} & \zeta_3^{(1)} & e_4 & \\ & e_1 & \delta_3^{(1)} & & & e_3 & \zeta_3^{(1)} & \beta_3^{(1)} & \zeta_4^{(1)} & \\ & & & \delta_4 & & & e_4 & \zeta_4^{(1)} & \beta_4 & \ddots \\ & & & & \ddots & & & & \ddots & \ddots \\ \hline \nu_1 & & & & & -\delta_1 & & & & \\ & \nu_2^{(1)} & e_2 & & & & -\delta_2^{(1)} & -e_1 & & \\ & e_2 & \nu_3^{(1)} & & & & -e_1 & -\delta_3^{(1)} & & \\ & & & \nu_4 & & & & & -\delta_4 & \\ & & & & \ddots & & & & & \ddots \end{array}\right]$$

with

$$\begin{array}{lcl} \delta_2^{(1)} &=& c_1^2\delta_2 + s_1^2\delta_3, \\ \delta_3^{(1)} &=& c_1^2\delta_3 + s_1^2\delta_2, \\ \nu_2^{(1)} &=& c_1^2\nu_2 + s_1^2\nu_3, \\ \nu_3^{(1)} &=& c_1^2\nu_3 + s_1^2\nu_2, \\ \zeta_2^{(1)} &=& c_1\zeta_2, \\ \zeta_4^{(1)} &=& c_1\zeta_4, \\ \zeta_3^{(1)} &=& c_1^2\zeta_3 + c_1 s_1(\beta_3 - \beta_2) - s_1^2\zeta_3, \\ \beta_2^{(1)} &=& c_1^2\beta_2 + 2c_1 s_1\zeta_3 + s_1^2\beta_3, \\ \beta_3^{(1)} &=& c_1^2\beta_3 - 2c_1 s_1\zeta_3 + s_1^2\beta_2. \end{array} \qquad \begin{array}{lcl} e_1 &=& c_1 s_1(\delta_3 - \delta_2), \\ e_2 &=& c_1 s_1(\nu_3 - \nu_2), \\ e_3 &=& -s_1\zeta_2, \\ e_4 &=& s_1\zeta_4, \end{array}$$

The introduction of the initial bulge is completed by $H_2 = \widetilde{G}_1^{(1)}\widetilde{G}_2^{(1)} H (\widetilde{G}_2^{(1)})^T (\widetilde{G}_1^{(1)})^T$,

$$H_2 = \left[\begin{array}{ccccc|ccccc} \delta_1^{(1)} & b_{10} & b_{11} & & & \beta_1^{(1)} & \zeta_2^{(2)} & b_7 & b_8 & \\ b_1 & \delta_2^{(2)} & b_{12} & & & \zeta_2^{(2)} & \beta_2^{(2)} & \zeta_3^{(2)} & b_9 & \\ b_2 & b_3 & \delta_3^{(1)} & & & b_7 & \zeta_3^{(2)} & \beta_3^{(1)} & \zeta_4^{(1)} & \\ & & & \delta_4 & & b_8 & b_9 & \zeta_4^{(1)} & \beta_4 & \ddots \\ & & & & \ddots & & & & \ddots & \ddots \\ \hline \nu_1^{(1)} & b_4 & b_5 & & & -\delta_1^{(1)} & -b_1 & -b_2 & & \\ b_4 & \nu_2^{(2)} & b_6 & & & -b_{10} & -\delta_2^{(2)} & -b_3 & & \\ b_5 & b_6 & \nu_3^{(1)} & & & -b_{11} & -b_{12} & -\delta_3^{(1)} & & \\ & & & \nu_4 & & & & & -\delta_4 & \\ & & & & \ddots & & & & & \ddots \end{array}\right]$$

with

$$
\begin{aligned}
\delta_1^{(1)} &= c_2^2\delta_1 + s_2^2\delta_2^{(1)}, & b_1 &= c_2 s_2(\delta_2^{(1)} - \delta_1),\\
\delta_2^{(2)} &= c_2^2\delta_2^{(1)} + s_2^2\delta_1, & b_2 &= s_2 e_1,\\
\nu_1^{(1)} &= c_2^2\nu_1 + s_2^2\nu_2^{(1)}, & b_3 &= c_2 e_1,\\
\nu_2^{(2)} &= c_2^2\nu_2^{(1)} + s_2^2\nu_1, & b_4 &= c_2 s_2(\nu_2^{(1)} - \nu_1),\\
\beta_1^{(1)} &= c_2^2\beta_1 + 2c_2 s_2\zeta_2^{(1)} + s_2^2\beta_2^{(1)}, & b_5 &= s_2 e_2,\\
\beta_2^{(2)} &= c_2^2\beta_2^{(1)} - 2c_2 s_2\zeta_2^{(1)} + s_1^2\beta_1, & b_6 &= c_2 e_2,\\
\zeta_2^{(2)} &= (c_2^2 - s_2^2)\zeta_2^{(1)} + c_2 s_2(\beta_2^{(1)} - \beta_1), & b_7 &= c_2 e_3 + s_2\zeta_3^{(1)},\\
\zeta_3^{(2)} &= c_2\zeta_3^{(1)} - s_2 b_3, & b_8 &= s_2 e_4,\\
& & b_9 &= c_2\zeta_3^{(1)} - s_2 e_3,\\
& & b_{10} &= b_1,\\
& & b_{11} &= b_2,\\
& & b_{12} &= b_3.
\end{aligned}
$$

At this point, the $(1,1)$ and consequently the $(2,2)$ block are symmetric, but this is not forced by the Hamiltonian structure and will soon be lost. Now we will chase the bulge, that is, we will restore the J-Hessenberg form by applying Algorithm 2.6, that is a symplectic matrix S_2 is constructed such that

$$\widetilde{H} = S_2 H_1 S_2^{-1} \tag{5.8}$$

is in J-Hessenberg form. As in the previous two sections, the algorithm simplifies due to the special structure of H_2. First the bulge b_5 is eliminated using a symplectic Givens transformation

$$
G_3^{(1)} = \left[\begin{array}{ccc|ccc}
I_2 & & & & & \\
 & c_3 & & & s_3 & \\
 & & I_{n-3} & & & \\
\hline
 & & & I_2 & & \\
 & -s_3 & & & c_3 & \\
 & & & & & I_{n-3}
\end{array}\right]
$$

with $[c_3, s_3] = \mathbf{givens}(b_2, b_5)$. This yields $H_3 = G_3^{(1)} H_2 (G_3^{(1)})^T$

$$
H_3 = \left[\begin{array}{cccccc|cccccc}
\delta_1^{(1)} & b_{10} & b_{11}^{(1)} & & & & \beta_1^{(1)} & \zeta_2^{(2)} & b_7^{(1)} & b_8 & & \\
b_1 & \delta_2^{(2)} & b_{12}^{(1)} & & & & \zeta_2^{(2)} & \beta_2^{(2)} & \zeta_3^{(3)} & b_9 & & \\
b_2^{(1)} & b_3^{(1)} & \delta_3^{(2)} & & & & b_7^{(1)} & \zeta_3^{(3)} & \beta_3^{(2)} & \zeta_4^{(2)} & & \\
 & & b_x & \delta_4 & & & b_8 & b_9 & \zeta_4^{(2)} & \beta_4 & \zeta_5 & \\
 & & & & \delta_5 & & & & & \zeta_5 & \beta_5 & \ddots \\
 & & & & & \ddots & & & & & \ddots & \ddots \\
\hline
\nu_1^{(1)} & b_4 & & & & & -\delta_1^{(1)} & -b_1 & -b_2^{(1)} & & & \\
b_4 & \nu_2^{(2)} & & & & & -b_{10} & -\delta_2^{(2)} & -b_3^{(1)} & & & \\
 & & \nu_3^{(2)} & & & & -b_{11}^{(1)} & -b_{12}^{(1)} & -\delta_3^{(2)} & -b_x & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \nu_5 & & & & & & -\delta_5 & \\
 & & & & & \ddots & & & & & & \ddots
\end{array}\right]
$$

with

$$\begin{aligned}
\delta_3^{(2)} &= (c_3^2 - s_3^2)\delta_3^{(1)} + c_3 s_3(\nu_3^{(1)} + \beta_3^{(1)}),\\
\nu_3^{(2)} &= c_3^2\nu_3^{(1)} - 2c_3 s_3\delta_3^{(1)} - s_3^2\beta_3^{(1)},\\
\beta_3^{(2)} &= c_3^2\beta_3^{(1)} - 2c_3 s_3\delta_3^{(1)} - s_3^2\nu_3^{(1)},\\
\zeta_3^{(3)} &= c_3\zeta_3^{(2)} - s_3 b_{12},\\
\zeta_4^{(2)} &= c_3\zeta_4^{(1)},\\
b_2^{(1)} &= c_3 b_2 + s_3 b_5,\\
b_3^{(1)} &= c_3 b_3 + s_3 b_6,\\
b_7^{(1)} &= c_3 b_7 - s_3 b_{11},\\
b_{11}^{(1)} &= c_3 b_{11} + s_3 b_7,\\
b_{12}^{(1)} &= c_3 b_{12} + s_3\zeta_3^{(2)},\\
b_x &= s_3\zeta_4^{(1)}.
\end{aligned}$$

Due to the choice of $G_3^{(1)}$ we have

$$\begin{aligned}
(H_3)_{n+3,1} &= 0\\
&= c_3 b_5 - s_3 b_2\\
&= c_3 s_2 e_2 - s_3 s_2 e_1\\
&= s_2(c_3 e_2 - s_3 e_1),
\end{aligned}$$

that is,

$$c_3 e_2 - s_3 e_1 = 0, \quad \text{or} \quad s_2 = 0.$$

If $s_2 = 0$, then $c_1 x_2 + s_1 x_3 = 0$ which implies that $x_2 = x_3 = 0$. This would imply $\nu_1\nu_2\zeta_2\zeta_3 = 0$, but as we are considering only unreduced Hamiltonian J-Hessenberg matrices, $\nu_j \neq 0$ for all $j = 1, \ldots, n$ and $\zeta_k \neq 0$ for all $k = 2, \ldots, n$. Therefore, $s_2 \neq 0$. Moreover,

$$\begin{aligned}
(H_3)_{n+3,2} &= c_3 b_6 - s_3 b_3\\
&= (H_3)_{n+2,3}\\
&= c_3 c_2 e_2 - s_3 c_2 e_1\\
&= c_2(c_3 e_2 - s_3 e_1)\\
&= 0.
\end{aligned}$$

Hence, this transformation generates not only one zero, but four of them. Two new entries are introduced. Moreover, the $(1,1)$ (and consequently the $(2,2)$) block is no longer symmetric. Next, a symplectic Givens transformation is used to eliminate the bulge element $b_4^{(1)}$

$$G_2^{(1)} = \left[\begin{array}{cc|cc}
1 & & & \\
 & c_4 & & s_4 \\
 & & I_{n-2} & \\
\hline
 & & 1 & \\
 & -s_4 & & c_4 \\
 & & & & I_{n-2}
\end{array}\right]$$

with $[c_4, s_4] = \mathbf{givens}(b_1, b_4)$. This yields $H_4 = G_2^{(1)} H_3 (G_2^{(1)})^T$

$$
H_4 = \left[\begin{array}{ccccccc|ccccccc}
\delta_1^{(1)} & b_{10}^{(1)} & b_{11}^{(1)} & & & & & \beta_1^{(1)} & \zeta_2^{(3)} & b_7^{(1)} & b_8 & & & \\
b_1^{(1)} & \delta_2^{(3)} & b_{12}^{(2)} & & & & & \zeta_2^{(3)} & \beta_2^{(3)} & \zeta_3^{(4)} & b_9^{(1)} & & & \\
b_2^{(1)} & b_3^{(2)} & \delta_3^{(2)} & & & & & b_7^{(1)} & \zeta_3^{(4)} & \beta_3^{(2)} & \zeta_4^{(2)} & & & \\
 & b_y & b_x & \delta_4 & & & & b_8 & b_9^{(1)} & \zeta_4^{(2)} & \beta_4 & \zeta_5 & & \\
 & & & & \delta_5 & & & & & & \zeta_5 & \beta_5 & \ddots & \\
 & & & & & \ddots & & & & & & \ddots & \ddots & \\
\hline
\nu_1^{(1)} & & & & & & & -\delta_1^{(1)} & -b_1^{(1)} & -b_2^{(1)} & & & & \\
 & \nu_2^{(3)} & b_4^{(1)} & & & & & -b_{10}^{(1)} & -\delta_2^{(3)} & -b_3^{(1)} & -b_y & & & \\
 & b_4^{(1)} & \nu_3^{(2)} & & & & & -b_{11}^{(1)} & -b_{12}^{(2)} & -\delta_3^{(2)} & -b_x & & & \\
 & & & \nu_4 & & & & & & & -\delta_4 & & & \\
 & & & & \nu_5 & & & & & & & -\delta_5 & & \\
 & & & & & \ddots & & & & & & & \ddots &
\end{array}\right]
$$

with

$$
\begin{aligned}
\delta_2^{(3)} &= (c_4^2 - s_4^2)\delta_2^{(2)} + c_4 s_4 (\nu_2^{(2)} + \beta_2^{(2)}), \\
\nu_2^{(3)} &= c_4^2 \nu_2^{(2)} - 2 c_4 s_4 \delta_2^{(2)} - s_4^2 \beta_2^{(2)}, \\
\beta_2^{(3)} &= c_4^2 \beta_2^{(2)} - 2 c_4 s_4 \delta_2^{(2)} - s_4^2 \nu_2^{(2)}, \\
\zeta_2^{(3)} &= c_4 \zeta_2^{(2)} - s_4 b_{10}, \\
\zeta_3^{(4)} &= c_4 \zeta_3^{(3)} - s_4 b_3^{(2)}, \\
b_1^{(1)} &= c_4 b_1 + s_4 b_4, \\
b_3^{(2)} &= c_4 b_3^{(1)} + s_4 \zeta_3^{(3)}, \\
b_4^{(1)} &= -s_4 b_{12}^{(1)}, \\
b_9^{(1)} &= c_4 b_9, \\
b_{10}^{(1)} &= c_4 b_{10} + s_4 \zeta_2^{(2)}, \\
b_{12}^{(2)} &= c_4 b_{12}^{(1)}, \\
b_y &= s_4 b_9.
\end{aligned}
$$

This transformation creates two additional entries. A symplectic Givens transformation of type II is used to eliminate $b_2^{(2)}$

$$
\widetilde{G}_2^{(2)} = \left[\begin{array}{cccc|cccc}
1 & & & & & & & \\
 & c_5 & s_5 & & & & & \\
 & -s_5 & c_5 & & & & & \\
 & & & I_{n-3} & & & & \\
\hline
 & & & & 1 & & & \\
 & & & & & c_5 & s_5 & \\
 & & & & & -s_5 & c_5 & \\
 & & & & & & & I_{n-3}
\end{array}\right],
$$

with $[c_5, s_5] = \mathbf{givens}(b_1^{(1)}, b_2^{(1)})$. This yields $H_5 = \widetilde{G}_2^{(2)} H_4 (\widetilde{G}_2^{(2)})^T$

$$
H_5 = \left[\begin{array}{cccccc|cccccc}
\delta_1^{(1)} & b_{10}^{(2)} & b_{11}^{(2)} & & & & \beta_1^{(1)} & \zeta_2^{(4)} & b_7^{(2)} & b_8 & & \\
b_1^{(2)} & \delta_2^{(4)} & b_{12}^{(3)} & & & & \zeta_2^{(4)} & \beta_2^{(4)} & \zeta_3^{(5)} & b_9^{(2)} & & \\
 & b_3^{(3)} & \delta_3^{(3)} & & & & b_7^{(2)} & \zeta_3^{(5)} & \beta_3^{(3)} & \zeta_4^{(3)} & & \\
 & b_y^{(1)} & b_x^{(1)} & \delta_4 & & & b_8 & b_9^{(2)} & \zeta_4^{(3)} & \beta_4 & \zeta_5 & \\
 & & & & \delta_5 & & & & & \zeta_5 & \beta_5 & \ddots \\
 & & & & & \ddots & & & & & \ddots & \ddots \\
\hline
\nu_1^{(1)} & & & & & & -\delta_1^{(1)} & -b_1^{(2)} & & & & \\
 & \nu_2^{(4)} & b_4^{(2)} & & & & -b_{10}^{(2)} & -\delta_2^{(4)} & -b_3^{(2)} & -b_y^{(1)} & & \\
 & b_4^{(2)} & \nu_3^{(3)} & & & & -b_{11}^{(2)} & -b_{12}^{(3)} & -\delta_3^{(3)} & -b_x^{(1)} & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \nu_5 & & & & & & -\delta_5 & \\
 & & & & & \ddots & & & & & & \ddots
\end{array}\right]
$$

with

$$
\begin{aligned}
\delta_2^{(4)} &= c_5^2 \delta_2^{(3)} + c_5 s_5 (b_3^{(2)} + b_{12}^{(2)}) + s_5^2 \delta_3^{(2)}, \\
\delta_3^{(3)} &= c_5^2 \delta_3^{(2)} - c_5 s_5 (b_{12}^{(2)} + b_3^{(2)}) + s_5^2 \delta_2^{(3)}, \\
\nu_2^{(4)} &= c_5^2 \nu_2^{(3)} + 2 c_5 s_5 b_4^{(1)} + s_5^2 \nu_3^{(2)}, \\
\nu_3^{(3)} &= c_5^2 \nu_3^{(2)} - 2 c_5 s_5 b_4^{(1)} + s_5^2 \nu_2^{(3)}, \\
\beta_2^{(4)} &= c_5^2 \beta_2^{(3)} + 2 c_5 s_5 \zeta_3^{(4)} + s_5^2 \beta_3^{(2)}, \\
\beta_3^{(3)} &= c_5^2 \beta_3^{(2)} - 2 c_5 s_5 \zeta_3^{(4)} + s_5^2 \beta_2^{(3)}, \\
\zeta_3^{(5)} &= (c_5^2 - s_5^2) \zeta_3^{(4)} + c_5 s_5 (\beta_3^{(2)} - \beta_2^{(3)}), \\
b_3^{(3)} &= c_5^2 b_3^{(2)} + c_5 s_5 (\delta_3^{(2)} - \delta_2^{(3)}) - s_5^2 b_{12}^{(2)}, \\
b_4^{(2)} &= (c_5^2 - s_5^2) b_4^{(1)} + c_5 s_5 (\nu_3^{(2)} - \nu_2^{(3)}), \\
b_{12}^{(3)} &= c_5^2 b_{12}^{(2)} + c_5 s_5 (\delta_3^{(2)} - \delta_2^{(3)}) - s_5^2 b_3^{(2)},
\end{aligned}
$$

and

$$
\begin{array}{lcllcl}
\zeta_2^{(4)} &=& c_5 \zeta_2^{(2)} + s_5 b_7^{(1)}, & b_1^{(2)} &=& c_5 b_1^{(1)} + s_5 b_2^{(1)}, \\
\zeta_4^{(3)} &=& c_5 \zeta_4^{(2)} - s_5 b_9^{(1)}, & b_7^{(2)} &=& c_5 b_7^{(1)} - s_5 \zeta_2^{(3)}, \\
b_9^{(2)} &=& c_5 b_9^{(1)} + s_5 \zeta_4^{(2)}, & b_{10}^{(2)} &=& c_5 b_{10}^{(1)} + s_5 b_{11}^{(1)}, \\
b_{11}^{(2)} &=& c_5 b_{11}^{(1)} - s_5 b_{10}^{(1)}, & b_x^{(1)} &=& c_5 b_x - s_5 b_y, \\
b_y^{(1)} &=& c_5 b_y + s_5 b_x. & & &
\end{array}
$$

No new matrix entries are created. Now a symplectic Gauss transformation L_1

$$
L_1 = \left[\begin{array}{ccc|ccc}
c_6 & & & & & d_6 \\
 & c_6 & & & d_6 & \\
 & & I_{n-2} & & & \\
\hline
 & & & c_6^{-1} & & \\
 & & & & c_6^{-1} & \\
 & & & & & I_{n-2}
\end{array}\right]
$$

where the parameters are chosen by $[c_6, d_6] = \mathbf{gauss1}(b_1^{(2)}, \nu_1^{(1)})$ in order to eliminate the $(2,1)$ element is applied, resulting in $H_6 = L_1 H_5 L_1^{-1}$

$$
H_6 = \left[\begin{array}{cccccc|cccccc}
\delta_1^{(1)} & b_{10}^{(3)} & b_{11}^{(3)} & & & & \beta_1^{(2)} & \zeta_2^{(5)} & b_7^{(3)} & b_8^{(1)} & & \\
 & \delta_2^{(4)} & b_{12}^{(4)} & & & & \zeta_2^{(5)} & \beta_2^{(5)} & \zeta_3^{(6)} & b_9^{(3)} & & \\
 & b_1^{(3)} & \delta_3^{(3)} & & & & b_7^{(3)} & \zeta_3^{(6)} & \beta_3^{(3)} & \zeta_4^{(3)} & & \\
 & b_2^{(2)} & b_3^{(4)} & \delta_4 & & & b_8^{(1)} & b_9^{(3)} & \zeta_4^{(3)} & \beta_4 & \zeta_5 & \\
 & & & & \delta_5 & & & & & \zeta_5 & \beta_5 & \ddots \\
 & & & & & \ddots & & & & & \ddots & \ddots \\
\hline
\nu_1^{(2)} & & & & & & -\delta_1^{(1)} & & & & & \\
 & \nu_2^{(5)} & b_4^{(3)} & & & & -b_{10}^{(3)} & -\delta_2^{(4)} & -b_1^{(3)} & -b_2^{(2)} & & \\
 & b_4^{(3)} & \nu_3^{(3)} & & & & -b_{11}^{(3)} & -b_{12}^{(4)} & -\delta_3^{(3)} & -b_3^{(4)} & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \nu_5 & & & & & & -\delta_5 & \\
 & & & & & \ddots & & & & & & \ddots
\end{array}\right]
$$

with

$$
\begin{aligned}
\nu_1^{(2)} &= c_6^{-2}\nu_1^{(1)}, \\
\nu_2^{(5)} &= c_6^{-2}\nu_2^{(4)}, \\
\beta_1^{(2)} &= c_6^2\beta_1^{(1)} - 2c_6 d_6 b_{10}^{(2)} - d_6^2\nu_2^{(4)}, \\
\beta_2^{(5)} &= c_6^2\beta_2^{(4)} - 2c_6 d_6 b_1^{(2)} - d_6^2\nu_1^{(1)}, \\
\zeta_2^{(5)} &= c_6^2\zeta_2^{(4)} - c_6 d_6(\delta_1^{(1)} + \delta_2^{(4)}), \\
\zeta_3^{(6)} &= c_6\zeta_3^{(5)} \\
b_1^{(3)} &= c_6^{-1}b_3^{(3)}, \\
b_2^{(2)} &= c_6^{-1}b_y^{(1)}, \\
b_3^{(4)} &= b_x^{(1)}, \\
b_4^{(3)} &= c_6^{-1}b_4^{(2)}, \\
b_7^{(3)} &= c_6 b_7^{(2)} - d_6 b_3^{(3)}, \\
b_8^{(1)} &= c_6 b_8 - d_6 b_y^{(1)}, \\
b_9^{(3)} &= c_6 b_9^{(2)}, \\
b_{10}^{(3)} &= b_{10}^{(2)} + c_6^{-1} d_6 \nu_2^{(4)}, \\
b_{11}^{(3)} &= c_6 b_{11}^{(2)} + d_6 b_4^{(2)}, \\
b_{12}^{(4)} &= c_6 b_{12}^{(3)}.
\end{aligned}
$$

No new entry is created. The bulge element $b_{11}^{(3)}$ is eliminated by a symplectic Givens

transformation $G_3^{(2)}$

$$G_3^{(2)} = \left[\begin{array}{ccc|ccc} I_2 & & & & & \\ & c_7 & & & s_7 & \\ & & I_{n-3} & & & \\ \hline & & & I_2 & & \\ & -s_7 & & & c_7 & \\ & & & & & I_{n-3} \end{array}\right]$$

with $[c_7, s_7] = \mathbf{givens}(b_7^{(3)}, -b_{11}^{(3)})$ This yields $H_7 = G_3^{(2)} H_6 (G_3^{(2)})^T$

$$H_7 = \left[\begin{array}{ccccccc|ccccccc} \delta_1^{(1)} & b_{10}^{(3)} & & & & & & \beta_1^{(2)} & \zeta_2^{(5)} & b_7^{(4)} & b_8^{(1)} & & & \\ & \delta_2^{(4)} & b_{12}^{(5)} & & & & & \zeta_2^{(5)} & \beta_2^{(5)} & \zeta_3^{(7)} & b_9^{(3)} & & & \\ & b_1^{(4)} & \delta_3^{(4)} & & & & & b_7^{(4)} & \zeta_3^{(7)} & \beta_3^{(4)} & \zeta_4^{(4)} & & & \\ & b_2^{(2)} & b_3^{(5)} & \delta_4 & & & & b_8^{(1)} & b_9^{(3)} & \zeta_4^{(4)} & \beta_4 & \zeta_5 & & \\ & & & & \delta_5 & & & & & & \zeta_5 & \beta_5 & \ddots & \\ & & & & & \ddots & & & & & & \ddots & \ddots & \\ \hline \nu_1^{(2)} & & & & & & & -\delta_1^{(1)} & & & & & & \\ & \nu_2^{(5)} & b_4^{(4)} & & & & & -b_{10}^{(3)} & -\delta_2^{(4)} & -b_1^{(4)} & -b_2^{(2)} & & & \\ & b_4^{(4)} & \nu_3^{(4)} & & & & & & -b_{12}^{(5)} & -\delta_3^{(4)} & -b_3^{(5)} & & & \\ & & & \nu_4 & & & & & & & -\delta_4 & & & \\ & & & & \nu_5 & & & & & & & -\delta_5 & & \\ & & & & & \ddots & & & & & & & \ddots & \end{array}\right]$$

with

$$\begin{array}{lcllcl} \delta_3^{(4)} & = & (c_7^2 - s_7^2)\delta_3^{(3)} + c_7 s_7(\nu_3^{(3)} + \beta_3^{(3)}), & b_1^{(4)} & = & c_7 b_1^{(3)} + s_7 b_4^{(3)}, \\ \nu_3^{(4)} & = & c_7^2 \nu_3^{(3)} - 2c_7 s_7 \delta_3^{(3)} - s_7^2 \beta_3^{(3)}, & b_3^{(5)} & = & c_7 b_3^{(4)} + s_7 \zeta_4^{(3)}, \\ \beta_3^{(4)} & = & c_7^2 \beta_3^{(3)} - 2c_7 s_7 \delta_3^{(3)} - s_7^2 \nu_3^{(3)}, & b_4^{(4)} & = & c_7 b_4^{(3)} - s_7 b_1^{(3)}, \\ \zeta_3^{(7)} & = & c_7 \zeta_3^{(6)} - s_7 b_{12}^{(4)}, & b_7^{(4)} & = & c_7 b_7^{(3)} - s_7 b_{11}^{(3)}, \\ \zeta_4^{(4)} & = & c_7 \zeta_4^{(3)} - s_7 b_3^{(4)}, & b_{12}^{(5)} & = & c_7 b_{12}^{(4)} + s_7 \zeta_3^{(6)}. \end{array}$$

No new entry is created. The bulge element $b_{10}^{(3)}$ is eliminated by a symplectic Givens transformation $G_2^{(2)}$

$$G_2^{(2)} = \left[\begin{array}{ccc|ccc} 1 & & & & & \\ & c_8 & & & s_8 & \\ & & I_{n-2} & & & \\ \hline & & & 1 & & \\ & -s_8 & & & c_8 & \\ & & & & & I_{n-2} \end{array}\right]$$

with $[c_8, s_8] =$ **givens**$(\zeta_2^{(5)}, -b_{10}^{(3)})$ This yields $H_8 = G_2^{(2)} H_7 (G_2^{(2)})^T$

$$H_8 = \left[\begin{array}{cccccc|cccccc}
\delta_1^{(1)} & & & & & & \beta_1^{(2)} & \zeta_2^{(6)} & b_7^{(4)} & b_8^{(1)} & & \\
 & \delta_2^{(5)} & b_{12}^{(6)} & & & & \zeta_2^{(6)} & \beta_2^{(6)} & \zeta_3^{(8)} & b_9^{(4)} & & \\
 & b_1^{(5)} & \delta_3^{(4)} & & & & b_7^{(4)} & \zeta_3^{(8)} & \beta_3^{(4)} & \zeta_4^{(4)} & & \\
 & b_2^{(3)} & b_3^{(5)} & \delta_4 & & & b_8^{(1)} & b_9^{(4)} & \zeta_4^{(4)} & \beta_4 & \zeta_5 & \\
 & & & & \delta_5 & & & & & \zeta_5 & \beta_5 & \ddots \\
 & & & & & \ddots & & & & & \ddots & \ddots \\
\hline
\nu_1^{(2)} & & & & & & -\delta_1^{(1)} & & & & & \\
 & \nu_2^{(6)} & b_4^{(5)} & & & & & -\delta_2^{(5)} & -b_1^{(4)} & -b_2^{(3)} & & \\
 & b_4^{(5)} & \nu_3^{(4)} & & & & & -b_{12}^{(6)} & -\delta_3^{(4)} & -b_3^{(5)} & & \\
 & & & \nu_4 & & & & & & -\delta_4 & & \\
 & & & & \nu_5 & & & & & & -\delta_5 & \\
 & & & & & \ddots & & & & & & \ddots
\end{array}\right]$$

with

$$\begin{array}{lcllcl}
\delta_2^{(5)} &=& (c_8^2 - s_8^2)\delta_2^{(4)} + c_8 s_8 (\nu_2^{(5)} + \beta_2^{(5)}), & b_1^{(5)} &=& c_8 b_1^{(4)} + s_8 \zeta_3^{(7)}, \\
\nu_2^{(6)} &=& c_8^2 \nu_2^{(5)} - 2 c_8 s_8 \delta_2^{(4)} - s_8^2 \beta_2^{(5)}, & b_2^{(3)} &=& c_8 b_2^{(2)} + s_8 b_9^{(3)}, \\
\beta_2^{(6)} &=& c_8^2 \beta_2^{(5)} - 2 c_8 s_8 \delta_2^{(4)} - s_8^2 \nu_2^{(5)}, & b_4^{(5)} &=& c_8 b_4^{(4)} - s_8 b_{12}^{(5)}, \\
\zeta_2^{(6)} &=& c_8 \zeta_2^{(5)} - s_8 b_{10}^{(3)}, & b_9^{(4)} &=& c_8 b_9^{(3)} - s_8 b_2^{(2)}, \\
\zeta_3^{(8)} &=& c_8 \zeta_3^{(7)} - s_8 b_1^{(4)}, & b_{12}^{(5)} &=& c_8 b_{12}^{(4)} + s_8 b_4^{(4)}.
\end{array}$$

Finally, the bulge entries $b_8^{(1)}$ and $b_7^{(4)}$ can be eliminated by either a symplectic Householder transformation or by the product of two symplectic Givens transformations of type II $\widetilde{G}_2^{(3)} \widetilde{G}_3^{(1)}$ where

$$\widetilde{G}_2^{(3)} = \left[\begin{array}{cccc|cccc}
1 & & & & & & & \\
 & c_{10} & s_{10} & & & & & \\
 & -s_{10} & c_{10} & & & & & \\
 & & & I_{n-3} & & & & \\
\hline
 & & & & 1 & & & \\
 & & & & & c_{10} & s_{10} & \\
 & & & & & -s_{10} & c_{10} & \\
 & & & & & & & I_{n-3}
\end{array}\right],$$

$$\widetilde{G}_3^{(1)} = \left[\begin{array}{cccc|cccc}
I_2 & & & & & & & \\
 & c_9 & s_9 & & & & & \\
 & -s_9 & c_9 & & & & & \\
 & & & I_{n-4} & & & & \\
\hline
 & & & & I_2 & & & \\
 & & & & & c_9 & s_9 & \\
 & & & & & -s_9 & c_9 & \\
 & & & & & & & I_{n-4}
\end{array}\right],$$

and $[c_9, s_9] =$ **givens**$(b_7^{(4)}, b_8^{(1)})$ and $[c_{10}, s_{10}] =$ **givens**$(\zeta_2^{(6)}, c_9 b_7^{(4)} + s_9 b_8^{(1)})$. Applying

$\widetilde{G}_3^{(1)}$ yields $H_9 = \widetilde{G}_3^{(1)} H_8 (\widetilde{G}_3^{(1)})^T$

$$
H_9 = \left[\begin{array}{ccccccc|ccccccc}
\delta_1^{(1)} & & & & & & & \beta_1^{(2)} & \zeta_2^{(6)} & b_7^{(5)} & & & & \\
 & \delta_2^{(5)} & b_{10}^{(4)} & b_{11}^{(4)} & & & & \zeta_2^{(6)} & \beta_2^{(6)} & \zeta_3^{(9)} & b_9^{(5)} & & & \\
 & b_1^{(6)} & \delta_3^{(5)} & b_{12}^{(7)} & & & & b_7^{(5)} & \zeta_3^{(9)} & \beta_3^{(5)} & \zeta_4^{(5)} & b_x^{(2)} & & \\
 & b_2^{(4)} & b_3^{(6)} & \delta_4^{(1)} & & & & & b_9^{(5)} & \zeta_4^{(5)} & \beta_4^{(1)} & \zeta_5^{(1)} & & \\
 & & & & \delta_5 & & & & & b_x^{(2)} & \zeta_5^{(1)} & \beta_5 & \ddots & \\
 & & & & & \ddots & & & & & & \ddots & \ddots & \\
\hline
\nu_1^{(2)} & & & & & & & -\delta_1^{(1)} & & & & & & \\
 & \nu_2^{(6)} & b_4^{(6)} & b_5^{(1)} & & & & & -\delta_2^{(5)} & -b_1^{(6)} & -b_2^{(4)} & & & \\
 & b_4^{(6)} & \nu_3^{(5)} & b_6^{(1)} & & & & & -b_{10}^{(4)} & -\delta_3^{(5)} & -b_3^{(6)} & & & \\
 & b_5^{(1)} & b_6^{(1)} & \nu_4^{(1)} & & & & & -b_{11}^{(4)} & -b_{12}^{(7)} & -\delta_4^{(1)} & & & \\
 & & & & \nu_5 & & & & & & & -\delta_5 & & \\
 & & & & & \ddots & & & & & & & \ddots &
\end{array}\right]
$$

with

$$
\begin{aligned}
\delta_3^{(5)} &= c_9^2 \delta_3^{(4)} + c_9 s_9 b_3^{(5)} + s_9^2 \delta_4, \\
\delta_4^{(1)} &= c_9^2 \delta_4 - c_9 s_9 b_3^{(5)} + s_9^2 \delta_3^{(4)}, \\
\nu_3^{(5)} &= c_9^2 \nu_3^{(4)} + s_9^2 \nu_4, \\
\nu_4^{(1)} &= c_9^2 \nu_4 + s_9^2 \nu_3^{(4)}, \\
\beta_3^{(5)} &= c_9^2 \beta_3^{(4)} + 2 c_9 s_9 \zeta_4^{(4)} + s_9^2 \beta_4, \\
\beta_4^{(1)} &= c_9^2 \beta_4 - 2 c_9 s_9 \zeta_4^{(4)} + s_9^2 \beta_3^{(4)}, \\
\zeta_3^{(9)} &= c_9 \zeta_3^{(8)} + s_9 b_9^{(4)}, \\
\zeta_4^{(5)} &= (c_9^2 - s_9^2) \zeta_4^{(4)} + c_9 s_9 (\beta_4 - \beta_3^{(4)}), \\
\zeta_5^{(1)} &= c_9 \zeta_5, \\
b_1^{(6)} &= c_9 b_1^{(5)} + s_9 b_2^{(3)}, \\
b_2^{(4)} &= c_9 b_2^{(3)} - s_9 b_1^{(5)}, \\
b_3^{(6)} &= c_9^2 b_3^{(5)} + c_9 s_9 (\delta_4 - \delta_3^{(4)}), \\
b_4^{(6)} &= c_9 b_4^{(5)}, \\
b_5^{(1)} &= -s_9 b_4^{(5)}, \\
b_6^{(1)} &= c_9 s_9 (\nu_4 - \nu_3^{(4)}) \\
b_7^{(5)} &= c_9 b_7^{(4)} + s_9 b_8^{(1)}, \\
b_9^{(5)} &= c_9 b_9^{(4)} - s_9 \zeta_3^{(8)}, \\
b_{10}^{(4)} &= c_9 b_{12}^{(6)}, \\
b_{11}^{(4)} &= -s_9 b_{12}^{(6)}, \\
b_{12}^{(7)} &= c_9 s_9 (\delta_4 - \delta_3^{(4)}) - s_9^2 b_3^{(5)}, \\
b_x^{(2)} &= s_9 \zeta_5.
\end{aligned}
$$

Applying $\widetilde{G}_2^{(3)}$ yields $H_{10} = \widetilde{G}_2^{(3)} H_9 (\widetilde{G}_2^{(3)})^T$

$$
\left[\begin{array}{ccccccc|ccccccc}
\delta_1^{(1)} & & & & & & & \beta_1^{(2)} & \zeta_2^{(7)} & & & & & \\
 & \delta_2^{(6)} & b_{10}^{(5)} & b_{11}^{(5)} & & & & \zeta_2^{(7)} & \beta_2^{(7)} & \zeta_3^{(10)} & b_7^{(6)} & b_8^{(2)} & & \\
 & b_1^{(7)} & \delta_3^{(6)} & b_{12}^{(8)} & & & & & \zeta_3^{(10)} & \beta_3^{(6)} & \zeta_4^{(6)} & b_9^{(6)} & & \\
 & b_2^{(5)} & b_3^{(7)} & \delta_4^{(1)} & & & & & b_7^{(6)} & \zeta_4^{(6)} & \beta_4^{(1)} & \zeta_5^{(1)} & & \\
 & & & & \delta_5 & & & & b_8^{(2)} & b_9^{(6)} & \zeta_5^{(1)} & \beta_5 & \ddots & \\
 & & & & & \ddots & & & & & & \ddots & \ddots & \\
\hline
\nu_1^{(2)} & & & & & & & -\delta_1^{(1)} & & & & & & \\
 & \nu_2^{(7)} & b_4^{(7)} & b_5^{(2)} & & & & & -\delta_2^{(6)} & -b_1^{(7)} & -b_2^{(5)} & & & \\
 & b_4^{(7)} & \nu_3^{(6)} & b_6^{(2)} & & & & & -b_{10}^{(5)} & -\delta_3^{(6)} & -b_3^{(7)} & & & \\
 & b_5^{(2)} & b_6^{(2)} & \nu_4^{(1)} & & & & & -b_{11}^{(5)} & -b_{12}^{(8)} & -\delta_4^{(1)} & & & \\
 & & & & \nu_5 & & & & & & & -\delta_5 & & \\
 & & & & & \ddots & & & & & & & \ddots &
\end{array}\right]
$$

with

$$
\begin{aligned}
\delta_2^{(6)} &= c_{10}^2\delta_2^{(5)} + c_{10}s_{10}(b_1^{(6)} + b_{10}^{(4)}) + s_{10}^2\delta_3^{(5)},\\
\delta_3^{(6)} &= c_{10}^2\delta_3^{(5)} - c_{10}s_{10}(b_1^{(6)} + b_{10}^{(4)}) + s_{10}^2\delta_2^{(5)},\\
\nu_2^{(7)} &= c_{10}^2\nu_2^{(6)} + 2c_{10}s_{10}b_4^{(6)} + s_{10}^2\nu_3^{(5)},\\
\nu_3^{(6)} &= c_{10}^2\nu_3^{(5)} - 2c_{10}s_{10}b_4^{(6)} + s_{10}^2\nu_2^{(6)},\\
\beta_2^{(7)} &= c_{10}^2\beta_2^{(6)} + 2c_{10}s_{10}\zeta_3^{(9)} + s_{10}^2\beta_3^{(5)},\\
\beta_3^{(6)} &= c_{10}^2\beta_3^{(5)} - 2c_{10}s_{10}\zeta_3^{(9)} + s_{10}^2\beta_2^{(6)},\\
\zeta_3^{(10)} &= (c_{10}^2 - s_{10})\zeta_3^{(9)} + c_{10}s_{10}(\beta_3^{(5)} - \beta_2^{(6)}),\\
b_1^{(7)} &= c_{10}^2 b_1^{(6)} + c_{10}s_{10}(\delta_3^{(5)} - \delta_2^{(5)}) - s_{10}^2 b_{10}^{(4)},\\
b_4^{(7)} &= (c_{10}^2 - s_{10}^2)b_4^{(6)} + c_{10}s_{10}(\nu_3^{(5)} - \nu_2^{(6)}),\\
b_{10}^{(5)} &= c_{10}^2 b_{10}^{(4)} + c_{10}s_{10}(\delta_3^{(5)} - \delta_2^{(5)}) - s_{10}^2 b_1^{(6)},
\end{aligned}
$$

and

$$
\begin{aligned}
\zeta_2^{(7)} &= c_{10}\zeta_2^{(6)} + s_{10}b_7^{(5)}, & b_2^{(5)} &= c_{10}b_2^{(4)} + s_{10}b_3^{(6)},\\
\zeta_4^{(6)} &= c_{10}\zeta_4^{(5)} - s_{10}b_9^{(5)}, & b_3^{(7)} &= c_{10}b_3^{(6)} - s_{10}b_2^{(5)},\\
b_5^{(2)} &= c_{10}b_5^{(1)} + s_{10}b_6^{(1)}, & b_6^{(2)} &= c_{10}b_6^{(1)} - s_{10}b_5^{(1)},\\
b_7^{(6)} &= c_{10}b_9^{(5)} + s_{10}\zeta_4^{(5)}, & b_8^{(2)} &= s_{10}b_x^{(2)},\\
b_9^{(6)} &= c_{10}b_x^{(2)}, & b_{11}^{(5)} &= c_{10}b_{11}^{(4)} + s_{10}b_{12}^{(7)},\\
b_{12}^{(8)} &= c_{10}b_{12}^{(7)} - s_{10}b_{11}^{(4)}.
\end{aligned}
$$

This is the situation as in H_2, just the bulge has moved one row and one column down in each of the blocks. Note, that the $(1,1)$ block was symmetric in H_2, here in H_{10} it is no longer symmetric. The entries $\delta_1^{(1)}, \beta_1^{(2)}$ and $\nu_1^{(2)}$ will not be changed in the rest of the computation, they already belong to the parameter set which determines $\widetilde{H}$ (5.8),

$$
\widetilde{\delta}_1 = \delta_1^{(1)}, \quad \widetilde{\beta}_1 = \beta_1^{(2)}, \quad \widetilde{\nu}_1 = \nu_1^{(2)}.
$$

As in the single and the double shift case, let us consider one more step of the bulge chasing process in order to derive an algorithm that works only on the parameters.

Next, the bulge $b_5^{(2)}$ is eliminate using a symplectic Givens transformation

$$G_4^{(1)} = \left[\begin{array}{ccc|ccc} I_3 & & & & & \\ & c_{11} & & & s_{11} & \\ & & I_{n-2} & & & \\ \hline & & & I_3 & & \\ & -s_{11} & & & c_{11} & \\ & & & & & I_{n-2} \end{array}\right]$$

with $[c_{11}, s_{11}] =$ **givens**$(b_2^{(5)}, b_5^{(2)})$. This yields $H_{11} = G_4^{(1)} H_{10} (G_4^{(1)})^T$

$$\left[\begin{array}{cccccc|cccccc} \widetilde{\delta}_1 & & & & & & \widetilde{\beta}_1 & \zeta_2^{(7)} & & & & \\ & \delta_2^{(6)} & b_{10}^{(5)} & b_{11}^{(6)} & & & \zeta_2^{(7)} & \beta_2^{(7)} & \zeta_3^{(10)} & b_7^{(7)} & b_8^{(2)} & \\ & b_1^{(7)} & \delta_3^{(6)} & b_{12}^{(9)} & & & & \zeta_3^{(10)} & \beta_3^{(6)} & \zeta_4^{(7)} & b_9^{(6)} & \\ & b_2^{(6)} & b_3^{(8)} & \delta_4^{(2)} & & & & b_7^{(7)} & \zeta_4^{(7)} & \beta_4^{(2)} & \zeta_5^{(2)} & \\ & & & b_x^{(3)} & \delta_5 & & & b_8^{(2)} & b_9^{(6)} & \zeta_5^{(2)} & \beta_5 & \ddots \\ & & & & & \ddots & & & & & \ddots & \ddots \\ \hline \widetilde{\nu}_1 & & & & & & -\widetilde{\delta}_1 & & & & & \\ & \nu_2^{(7)} & b_4^{(7)} & & & & & -\delta_2^{(6)} & -b_1^{(7)} & -b_2^{(6)} & & \\ & b_4^{(7)} & \nu_3^{(6)} & b_6^{(3)} & & & & -b_{10}^{(5)} & -\delta_3^{(6)} & -b_3^{(8)} & & \\ & & b_6^{(3)} & \nu_4^{(2)} & & & & -b_{11}^{(6)} & -b_{12}^{(9)} & -\delta_4^{(2)} & -b_x^{(3)} & \\ & & & & \nu_5 & & & & & & -\delta_5 & \\ & & & & & \ddots & & & & & & \ddots \end{array}\right]$$

with

$$\begin{aligned} \delta_4^{(2)} &= (c_{11}^2 - s_{11}^2)\delta_4^{(1)} + c_{11}s_{11}(\nu_4^{(1)} + \beta_4^{(1)}), \\ \nu_4^{(2)} &= c_{11}^2\nu_4^{(1)} - 2c_{11}s_{11}\delta_4^{(1)} - s_{11}^2\beta_4^{(1)}, \\ \beta_4^{(2)} &= c_{11}^2\beta_4^{(1)} - 2c_{11}s_{11}\delta_4^{(1)} - s_{11}^2\nu_4^{(1)}, \\ \zeta_4^{(7)} &= c_{11}\zeta_4^{(6)} - s_{11}b_{12}^{(8)}, \\ \zeta_5^{(2)} &= c_{11}\zeta_5^{(1)}, \end{aligned} \qquad \begin{aligned} b_2^{(6)} &= c_{11}b_2^{(5)} + s_{11}b_5^{(2)}, \\ b_3^{(8)} &= c_{11}b_3^{(7)} + s_{11}b_6^{(2)}, \\ b_6^{(3)} &= c_{11}b_6^{(2)} - s_{11}b_3^{(7)}, \\ b_7^{(1)} &= c_{11}b_7^{(6)} - s_{11}b_{11}^{(5)}, \\ b_{11}^{(6)} &= c_{11}b_{11}^{(5)} + s_{11}b_7^{(6)}, \\ b_{12}^{(9)} &= c_{11}b_{12}^{(8)} + s_{11}\zeta_4^{(6)}, \\ b_x^{(3)} &= s_{11}\zeta_5^{(1)}. \end{aligned}$$

All the computations are the same as for computing H_3, just the indices of the parameters $\delta, \nu, \beta, \zeta$ have to be increased by one. While in H_3 the bulge entry, due to the special problem set up, $b_6^{(1)}$ was zero, here $b_6^{(3)} \neq 0$. Hence, we need to be careful in the next reduction step as $b_6^{(3)}$ will enter the computation and alter the formula we obtained for H_4 slightly.

Next, a symplectic Givens transformation is used to eliminate the bulge element

$b_4^{(7)}$

$$G_3^{(3)} = \left[\begin{array}{ccc|ccc} I_2 & & & & & \\ & c_{12} & & & s_{12} & \\ & & I_{n-3} & & & \\ \hline & & & I_2 & & \\ & -s_{12} & & & c_{12} & \\ & & & & & I_{n-3} \end{array}\right]$$

with $[c_{12}, s_{12}] = \mathbf{givens}(b_1^{(7)}, b_4^{(7)})$. This yields $H_{12} = G_3^{(3)} H_{11} (G_3^{(3)})^T$

$$\left[\begin{array}{ccccccc|ccccccc} \widetilde{\delta}_1 & & & & & & & \widetilde{\beta}_1 & \zeta_2^{(7)} & & & & & \\ & \delta_2^{(6)} & b_{10}^{(6)} & b_{11}^{(6)} & & & & \zeta_2^{(7)} & \beta_2^{(7)} & \zeta_3^{(11)} & b_7^{(7)} & b_8^{(2)} & & \\ & b_1^{(8)} & \delta_3^{(7)} & b_{12}^{(10)} & & & & & \zeta_3^{(11)} & \beta_3^{(7)} & \zeta_4^{(8)} & b_9^{(7)} & & \\ & b_2^{(6)} & b_3^{(9)} & \delta_4^{(2)} & & & & & b_7^{(7)} & \zeta_4^{(8)} & \beta_4^{(2)} & \zeta_5^{(2)} & & \\ & & b_y^{(2)} & b_x^{(3)} & \delta_5 & & & & b_8^{(2)} & b_9^{(7)} & \zeta_5^{(2)} & \beta_5 & \ddots & \\ & & & & & \ddots & & & & & & \ddots & \ddots & \\ \hline \widetilde{\nu}_1 & & & & & & & -\widetilde{\delta}_1 & & & & & & \\ & \nu_2^{(7)} & & & & & & & -\delta_2^{(6)} & -b_1^{(8)} & -b_2^{(6)} & & & \\ & & \nu_3^{(7)} & b_4^{(8)} & & & & & -b_{10}^{(6)} & -\delta_3^{(7)} & -b_3^{(9)} & -b_y^{(2)} & & \\ & & b_4^{(8)} & \nu_4^{(3)} & & & & & -b_{11}^{(6)} & -b_{12}^{(10)} & -\delta_4^{(2)} & -b_x^{(3)} & & \\ & & & & \nu_5 & & & & & & & -\delta_5 & & \\ & & & & & \ddots & & & & & & & \ddots & \end{array}\right]$$

with

$$\begin{array}{lcl} \delta_3^{(7)} &=& (c_{12}^2 - s_{12}^2)\delta_3^{(6)} + c_{12}s_{12}(\nu_3^{(6)} + \beta_3^{(6)}), \\ \nu_3^{(6)} &=& c_{12}^2\nu_3^{(6)} - 2c_{12}s_{12}\delta_3^{(6)} - s_{12}^2\beta_3^{(6)}, \\ \beta_3^{(7)} &=& c_{12}^2\beta_3^{(6)} - 2c_{12}s_{12}\delta_3^{(6)} - s_{12}^2\nu_3^{(6)}, \\ \zeta_3^{(11)} &=& c_{12}\zeta_3^{(10)} - s_{12}b_{10}^{(5)}, \\ \zeta_4^{3)} &=& c_{12}\zeta_4^{(2)} - s_{12}b_3^{(8)}, \end{array} \qquad \begin{array}{lcl} b_1^{(8)} &=& c_{12}b_1^{(7)} + s_{12}b_4^{(7)}, \\ b_3^{(9)} &=& c_{12}b_3^{(8)} + s_{12}\zeta_4^{(7)}, \\ b_4^{(8)} &=& c_{12}b_6^{(3)} - s_{12}b_{12}^{(9)}, \\ b_9^{(7)} &=& c_{12}b_9^{(8)}, \\ b_{10}^{(6)} &=& c_{12}b_{10}^{(5)} + s_{12}\zeta_3^{(10)}, \\ b_{12}^{(10)} &=& c_{12}b_{12}^{(10)} + s_{12}b_6^{(3)}, \\ b_y &=& s_{12}b_9^{(8)}. \end{array}$$

As before, this transformation creates two additional entries. The computations of the bulge elements b_4 and b_{12} differ from the computation of H_4 due to the fact that in the previous step, b_6 did not vanish.

A symplectic Givens transformation of type II is used to eliminate $b_2^{(6)}$

$$\widetilde{G}_3^{(1)} = \left[\begin{array}{ccccc|ccccc} I_2 & & & & & & & & & \\ & c_{13} & s_{13} & & & & & & & \\ & -s_{13} & c_{13} & & & & & & & \\ & & & I_{n-4} & & & & & & \\ \hline & & & & & I_2 & & & & \\ & & & & & & c_{13} & s_{13} & & \\ & & & & & & -s_{13} & c_{13} & & \\ & & & & & & & & & I_{n-4} \end{array}\right],$$

with $[c_{13}, s_{13}] = \mathbf{givens}(b_1^{(8)}, b_2^{(6)})$. This yields $H_{13} = \widetilde{G}_3^{(1)} H_{12} (\widetilde{G}_3^{(1)})^T$

$$\left[\begin{array}{ccccc|ccccc}
\widetilde{\delta}_1 & & & & & \widetilde{\beta}_1 & \zeta_2^{(7)} & & & \\
 & \delta_2^{(6)} & b_{10}^{(7)} & b_{11}^{(7)} & & \zeta_2^{(7)} & \beta_2^{(7)} & \zeta_3^{(12)} & b_7^{(8)} & b_8^{(2)} \\
 & b_1^{(9)} & \delta_3^{(8)} & b_{12}^{(11)} & & & \zeta_3^{(12)} & \beta_3^{(8)} & \zeta_4^{(9)} & b_9^{(8)} \\
 & & b_3^{(10)} & \delta_4^{(3)} & & & b_7^{(8)} & \zeta_4^{(9)} & \beta_4^{(3)} & \zeta_5^{(3)} \\
 & & b_y^{(3)} & b_x^{(4)} & \delta_5 & & b_8^{(2)} & b_9^{(8)} & \zeta_5^{(3)} & \beta_5 \\
 & & & & \ddots & & & & \ddots & \ddots \\
\hline
\widetilde{\nu}_1 & & & & & -\widetilde{\delta}_1 & & & & \\
 & \nu_2^{(7)} & & & & & -\delta_2^{(6)} & -b_1^{(9)} & & \\
 & & \nu_3^{(8)} & b_4^{(9)} & & & -b_{10}^{(7)} & -\delta_3^{(8)} & -b_3^{(10)} & -b_y^{(3)} \\
 & & b_4^{(9)} & \nu_4^{(4)} & & & -b_{11}^{(7)} & -b_{12}^{(11)} & -\delta_4^{(3)} & -b_x^{(4)} \\
 & & & & \nu_5 & & & & & -\delta_5 \\
 & & & & \ddots & & & & & \ddots
\end{array}\right]$$

with

$$\begin{aligned}
\delta_3^{(8)} &= c_{13}^2 \delta_3^{(7)} + c_{13} s_{13} (b_3^{(9)} + b_{12}^{(10)}) + s_{13}^2 \delta_4^{(2)}, \\
\delta_4^{(3)} &= c_{13}^2 \delta_4^{(2)} - c_{13} s_{13} (b_{12}^{(10)} + b_3^{(9)}) + s_{13}^2 \delta_3^{(7)}, \\
\nu_3^{(8)} &= c_{13}^2 \nu_3^{(7)} + 2 c_{13} s_{13} b_4^{(8)} + s_{13}^2 \nu_4^{(3)}, \\
\nu_4^{(4)} &= c_{13}^2 \nu_4^{(3)} - 2 c_{13} s_{13} b_4^{(8)} + s_{13}^2 \nu_3^{(7)}, \\
\beta_3^{(8)} &= c_{13}^2 \beta_3^{(7)} + 2 c_{13} s_{13} \zeta_4^{(8)} + s_{13}^2 \beta_4^{(2)}, \\
\beta_4^{(3)} &= c_{13}^2 \beta_4^{(2)} - 2 c_{13} s_{13} \zeta_4^{(8)} + s_{13}^2 \beta_3^{(7)}, \\
\zeta_4^{(9)} &= (c_{13}^2 - s_{13}^2) \zeta_4^{(8)} + c_{13} s_{13} (\beta_4^{(2)} - \beta_3^{(7)}), \\
b_3^{(10)} &= c_{13}^2 b_3^{(9)} + c_{13} s_{13} (\delta_4^{(2)} - \delta_3^{(7)}) - s_{13}^2 b_{12}^{(10)}, \\
b_4^{(9)} &= (c_{13}^2 - s_{13}^2) b_4^{(8)} + c_{13} s_{13} (\nu_4^{(3)} - \nu_3^{(7)}), \\
b_{12}^{(11)} &= c_{13}^2 b_{12}^{(10)} + c_{13} s_{13} (\delta_4^{(2)} - \delta_3^{(7)}) - s_{13}^2 b_3^{(9)},
\end{aligned}$$

and

$$\begin{array}{lcllcl}
\zeta_3^{(12)} &=& c_{13} \zeta_3^{(11)} + s_{13} b_7^{(7)}, & b_1^{(9)} &=& c_{13} b_1^{(8)} + s_{13} b_2^{(6)}, \\
\zeta_5^{(3)} &=& c_{13} \zeta_5^{(2)} - s_{13} b_9^{(7)}, & b_7^{(8)} &=& c_{13} b_7^{(7)} - s_{13} \zeta_3^{(11)}, \\
b_9^{(8)} &=& c_{13} b_9^{(7)} + s_{13} \zeta_5^{(2)}, & b_{10}^{(7)} &=& c_{13} b_{10}^{(6)} + s_{13} b_{11}^{(6)}, \\
b_{11}^{(7)} &=& c_{13} b_{11}^{(6)} - s_{13} b_{10}^{(16}, & b_x^{(4)} &=& c_{13} b_x^{(3)} - s_{13} b_y^{(2)}, \\
b_y^{(3)} &=& c_{13} b_y^{(2)} + s_{13} b_x^{(3)}. & & &
\end{array}$$

No new matrix entries are created. The computation is just as in the computation of H_5, only the indices of all parameters have to be increased by one.

Now a symplectic Gauss transformation L_2

$$L_2 = \left[\begin{array}{ccccc|ccccc} 1 & & & & & & & & & \\ & c_{14} & & & & & & & d_{14} & \\ & & c_{14} & & & & & d_{14} & & \\ & & & I_{n-3} & & & & & & \\ \hline & & & & & 1 & & & & \\ & & & & & & c_{14}^{-1} & & & \\ & & & & & & & c_{14}^{-1} & & \\ & & & & & & & & I_{n-3} & \end{array}\right]$$

where the parameters are chosen by $[c_{14}, d_{14}] =$ **gauss1**$(b_1^{(9)}, \nu_2^{(7)})$ in order to eliminate the $(2,1)$ element is applied, resulting in $H_{14} = L_2 H_{13} L_2^{-1}$

$$\left[\begin{array}{ccccc|ccccc} \widetilde{\delta}_1 & & & & & \widetilde{\beta}_1 & \zeta_2^{(8)} & & & \\ & \delta_2^{(6)} & b_{10}^{(8)} & b_{11}^{(8)} & & \zeta_2^{(8)} & \beta_2^{(8)} & \zeta_3^{(13)} & b_7^{(9)} & b_8^{(3)} \\ & & \delta_3^{(8)} & b_{12}^{(12)} & & & \zeta_3^{(13)} & \beta_3^{(9)} & \zeta_4^{(10)} & b_9^{(9)} \\ & & b_1^{(10)} & \delta_4^{(3)} & & & b_7^{(9)} & \zeta_4^{(10)} & \beta_4^{(3)} & \zeta_5^{(3)} \\ & & b_2^{(7)} & b_3^{(11)} & \delta_5 & & b_8^{(3)} & b_9^{(9)} & \zeta_5^{(3)} & \beta_5 \\ & & & & \ddots & & & & & \ddots \\ \hline \widetilde{\nu}_1 & & & & & -\widetilde{\delta}_1 & & & & \\ & \nu_2^{(8)} & & & & & -\delta_2^{(6)} & & & \\ & & \nu_3^{(9)} & b_4^{(10)} & & & -b_{10}^{(8)} & -\delta_3^{(8)} & -b_1^{(10)} & -b_2^{(7)} \\ & & b_4^{(10)} & \nu_4^{(5)} & & & -b_{11}^{(8)} & -b_{12}^{(12)} & -\delta_4^{(3)} & -b_3^{(11)} \\ & & & & \nu_5 & & & & & -\delta_5 \\ & & & & \ddots & & & & & \end{array}\right]$$

with

$$\begin{aligned} \nu_2^{(8)} &= c_{14}^{-2}\nu_2^{(7)}, \\ \nu_3^{(9)} &= c_{14}^{-2}\nu_3^{(8)}, \\ \beta_2^{(8)} &= c_{14}^2\beta_2^{(7)} - 2c_{14}d_{14}b_{10}^{(7)} - d_{14}^2\nu_3^{(8)}, \\ \beta_3^{(9)} &= c_{14}^2\beta_3^{(8)} - 2c_{14}d_{14}b_1^{(9)} - d_{14}^2\nu_2^{(7)}, \\ \zeta_2^{(8)} &= c_{14}\zeta_2^{(7)}, \\ \zeta_3^{(13)} &= c_{14}^2\zeta_3^{(12)} - c_{14}d_{14}(\delta_2^{(6)} + \delta_3^{(8)}), \\ \zeta_4^{(10)} &= c_{14}\zeta_4^{(9)} \end{aligned}$$

$$\begin{aligned} b_1^{(10)} &= c_{14}^{-1}b_3^{(10)}, \\ b_2^{(7)} &= c_{14}^{-1}b_y^{(3)}, \\ b_3^{(11)} &= b_x^{(1)}, \\ b_4^{(10)} &= c_{14}^{-1}b_4^{(9)}, \\ b_7^{(9)} &= c_{14}b_7^{(8)} - d_{14}b_3^{(10)}, \\ b_8^{(3)} &= c_{14}b_8^{(2)} - d_{14}b_y^{(3)}, \\ b_9^{(9)} &= c_{14}b_9^{(8)}, \\ b_{10}^{(8)} &= b_{10}^{(7)} + c_{14}^{-1}d_{14}\nu_3^{(8)}, \\ b_{11}^{(8)} &= c_{14}b_{11}^{(7)} + d_{14}b_4^{(9)}, \\ b_{12}^{(12)} &= c_{14}b_{12}^{(11)}. \end{aligned}$$

No new entry is created. The only difference to the computation of H_6 is (upto increasing the indices of all parameters by one) the change of the parameter ζ_2.

The final 4 transformations which ensure that the bulge will move one row and column further down the diagonal in each of the four blocks do not reveal any new computations, therefore we refrain from stating them explicitly. After these transformations, the bulge has moved further down, the parameters $\widetilde{\delta}_2, \widetilde{\beta}_2, \widetilde{\zeta}_2$ and $\widetilde{\nu}_2$ of the final $\widetilde{H}$ can be read off.

From the given reduction it is easy to derive an algorithm that computes the parameters of $\widetilde{H}$ one set (that is, $\widetilde{\delta}_{j+1}, \widetilde{\beta}_j, \widetilde{\zeta}_j, \widetilde{\nu}_j$) at a time given the parameters of H. In order to do so, take all the formulae needed for the computing of H_{11}, H_{12}, H_{13}, and H_{14} and change the indices of the parameters in the following way

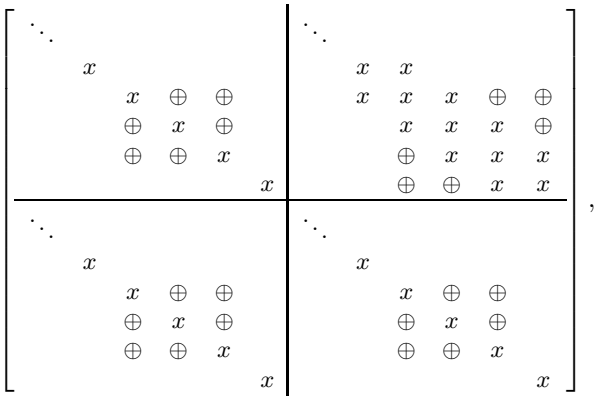
$$3 \to j+1, \quad 4 \to j+2, \quad 5 \to j+3,$$

the indices of the bulge entries do not change. Next take all the formulae needed for the computing of H_7, H_8, H_9, and H_{10} and change the indices of the parameters in the following way

$$2 \to j+1, \quad 3 \to j+2, \quad 4 \to j+3,$$

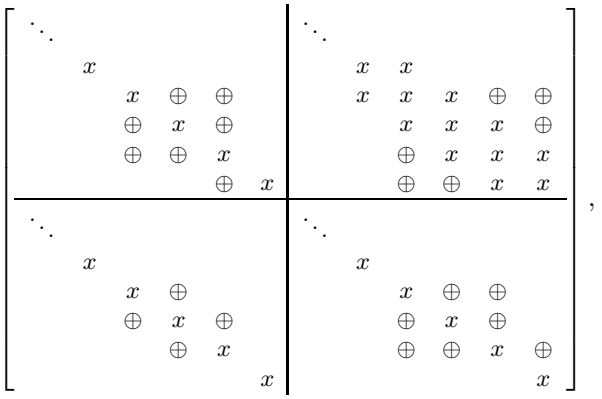
as before, the indices of the bulge entries do not change.

One has to be careful when the final columns are treated, as there is no ζ_{n+1} and the bulge will be chased out of the matrix. Assume that we have achieved

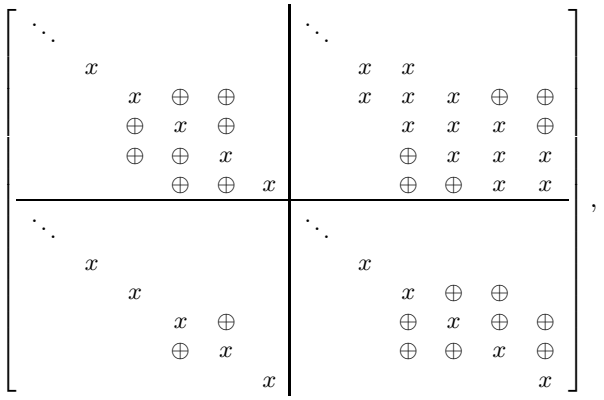


where for notional simplicity the bulge entries are denoted by $\oplus$ and all other entries by x. The parameterized algorithm for moving the bulge obtained above applies the following sequence of transformations:

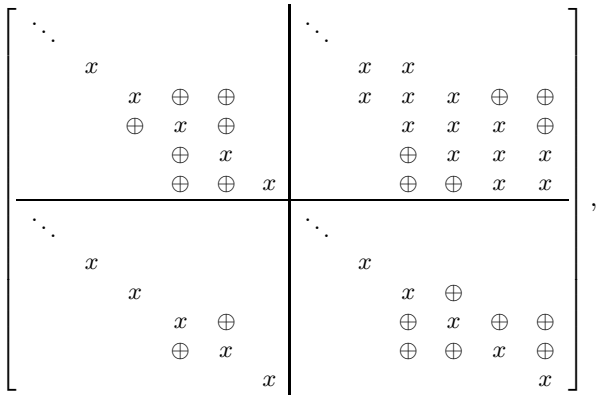
- a symplectic Givens transformation G_{n-1} to eliminate the entry $(2n-1, n-3)$



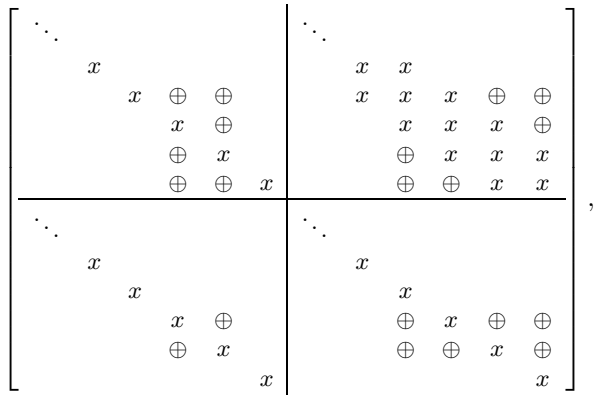
- a symplectic Givens transformation G_{n-2} to eliminate the entry $(2n-2, n-3)$



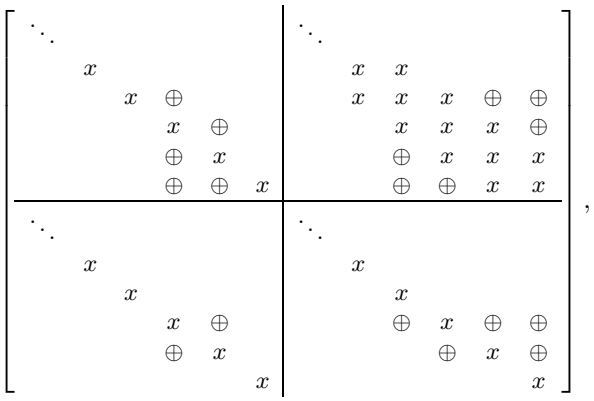
- a symplectic Givens transformation of type II $\widetilde{G}_{n-2}$ to eliminate the entry $(n-1, n-3)$



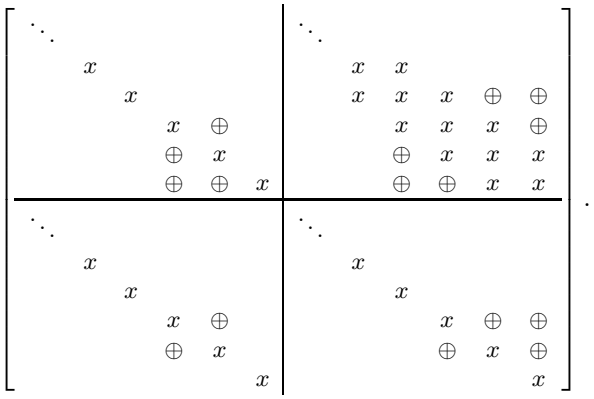
- a symplectic Gauss transformation L_{n-2} to eliminate the entry $(n-2, n-3)$



- a symplectic Givens transformation G_{n-1} to eliminate the entry $(2n-1, 2n-3)$

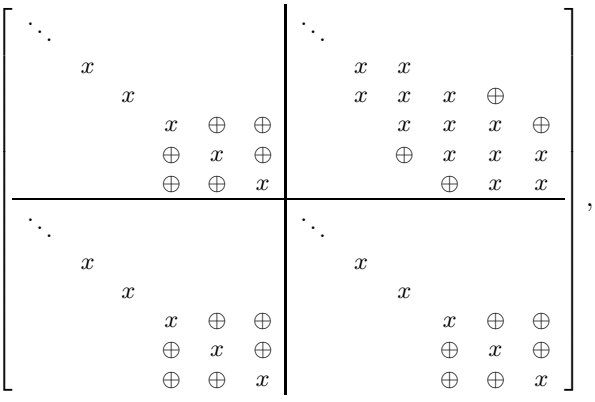


- a symplectic Givens transformation G_{n-2} to eliminate the entry $(2n-2, 2n-3)$



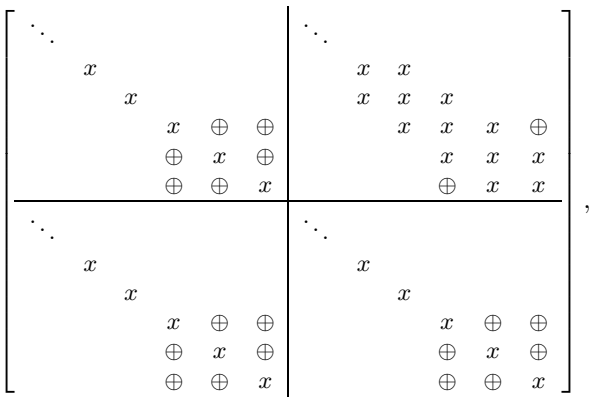
All of these computations are identical to the previously derived formulae. The final two transformation generate the 'next' bulge, some of the bulge entries already 'leave' the matrix, moreover there is no ζ_{n+1}. Therefore, when applying

- a symplectic Givens transformation of type II $\widetilde{G}_{n-1}$ to eliminate the entry $(n, n-3)$



there are no b_x and ζ_{n+1},

- while when applying a symplectic Givens transformation of type II $\widetilde{G}_{n-2}$ to eliminate the entry $(n-1, n-3)$

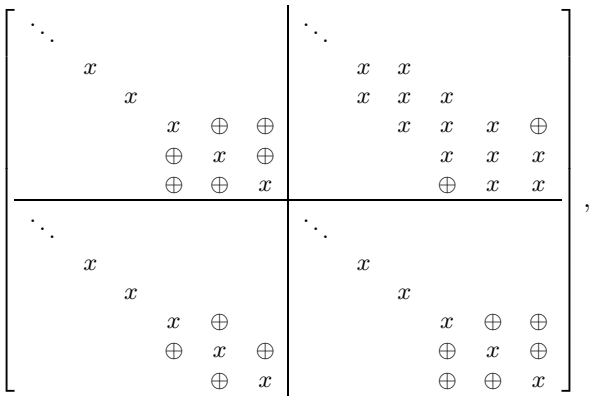


there are no b_8 and b_9.

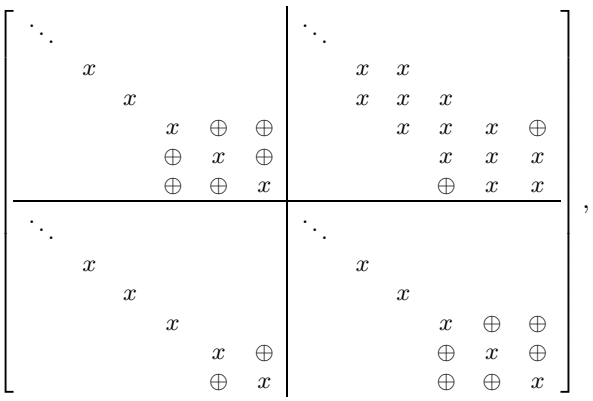
The bulge has moved one row and one column further down. As it is been chased out of the matrix, its size has already decreased by 4 elements. In all subsequent computations the computation of b_8, b_9 and ζ_{n+1} is not necessary.

For the next iteration step, we list the transformations and which bulge entries 'leave' the matrix and do not need to be considered further. Following the general derivation of an implicit quadruple SR step as described above the following sequence of transformation is applied

- a symplectic Givens transformation G_n to eliminate the entry $(2n, n-2)$

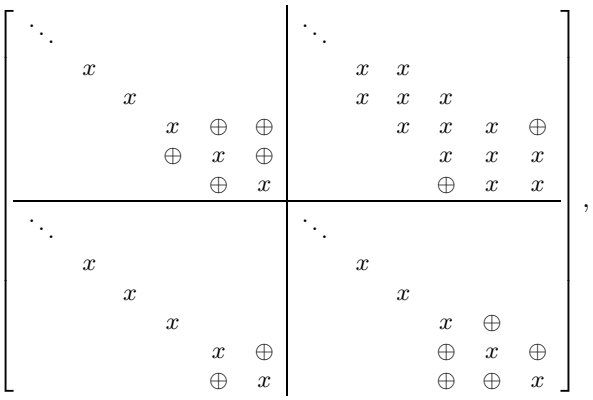


- a symplectic Givens transformation G_{n-1} to eliminate the entry $(2n-1, n-2)$

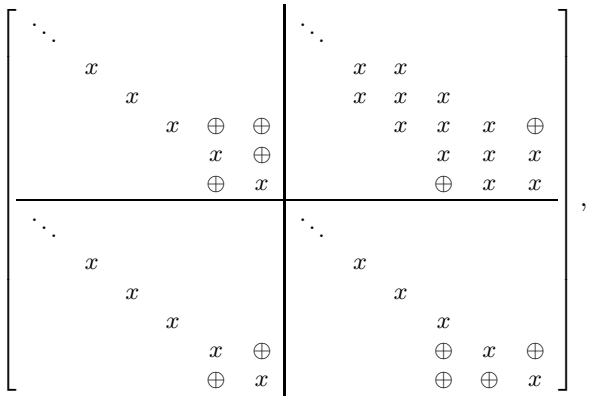


there are no b_y and b_9,

- a symplectic Givens transformation of type II G_{n-1} to eliminate the entry $(n, n-2)$

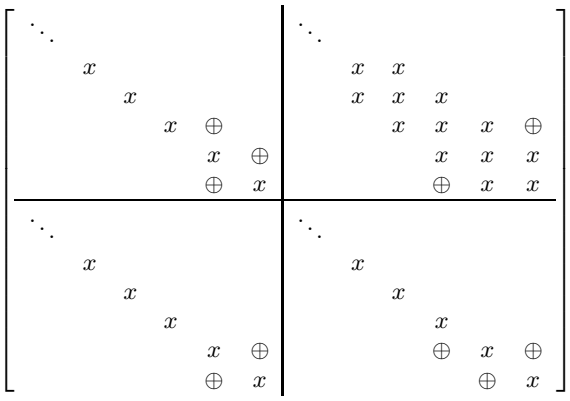


- a symplectic Gauss transformation L_{n-1} to eliminate the entry $(n-1, n-2)$

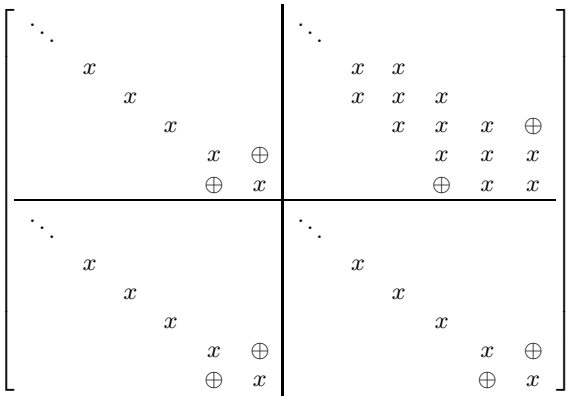


there are no b_2, b_3, and b_8

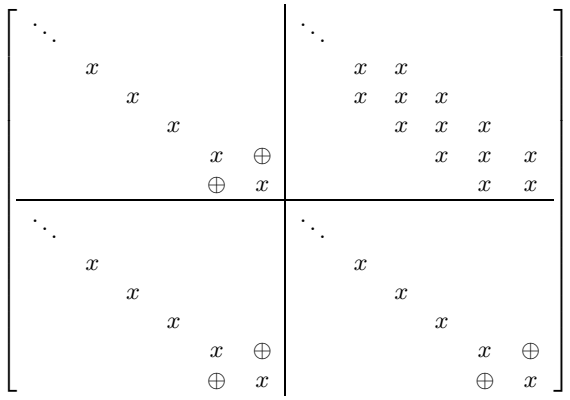
- a symplectic Givens transformation G_n to eliminate the entry $(2n, 2n-2)$



- a symplectic Givens transformation G_{n-1} to eliminate the entry $(2n-1, 2n-2)$



- a transformation $\widetilde{G}_n$ is not needed, therefore in the next step we have to be careful as the missing transformation corresponds to the one generating H_9 from H_8 which relabels the bulge entries,
- a symplectic Givens transformation of type II $\widetilde{G}_{n-1}$ to eliminate the entry $(n, 2n-2)$



  there are no $b_5, b_6, b_7, b_{11}, b_{12}$, and δ_{n+1},
- a transformation G_{n+1} is not needed.

The bulge has been moved once again. It consists of only four elements. Note that the $(1,2)$-block is already in the desired form. In order to account for the missing relabeling of the bulge entries, the last transformation performed has to be modified as follows: we have obtained

$$\left[\begin{array}{cccc|ccccc}
\ddots & & & & \ddots & & & & \\
 & \delta_{n-2} & & & & \beta_{n-2} & \zeta_{n-1} & b_7 & \\
 & & \delta_{n-1} & b_{12} & & \zeta_{n+1} & \beta_{n-1} & \zeta_n & \\
 & & b_1 & \delta_n & & b_7 & \zeta_n & \beta_n & \\
\hline
\ddots & & & & \ddots & & & & \\
 & \nu_{n-2} & & & & -\delta_{n-2} & & & \\
 & & \nu_{n-1} & b_4 & & & -\delta_{n-1} & -b_1 & \\
 & & b_4 & \nu_n & & & -b_{12} & -\delta_n &
\end{array}\right],$$

applying the transformation $\widetilde{G}_{n-1}$ gives

$$\left[\begin{array}{cccc|ccccc}
\ddots & & & & \ddots & & & & \\
 & \delta_{n-2} & & & & \beta_{n-2} & \zeta_{n-1} & & \\
 & & \delta_{n-1} & b_{10} & & \zeta_{n+1} & \beta_{n-1} & \zeta_n & \\
 & & b_1 & \delta_n & & & \zeta_n & \beta_n & \\
\hline
\ddots & & & & \ddots & & & & \\
 & \nu_{n-2} & & & & -\delta_{n-2} & & & \\
 & & \nu_{n-1} & b_4 & & & -\delta_{n-1} & -b_1 & \\
 & & b_4 & \nu_n & & & -b_{10} & -\delta_n &
\end{array}\right],$$

where the formulae for b_1, b_{10} and δ_{n-1} have to be modified slightly, all other entries change as before.

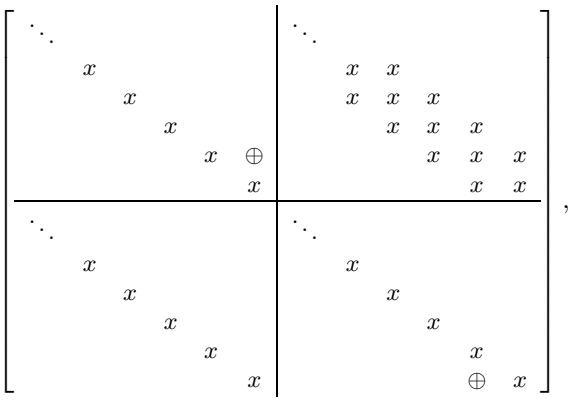
Following the general derivation of an implicit quadruple SR step as described above the following sequence of transformation is applied to drive the bulge out of the matrix

- a transformation G_n is not needed,
- a symplectic Givens transformation G_n to eliminate the entry $(2n, n-1)$

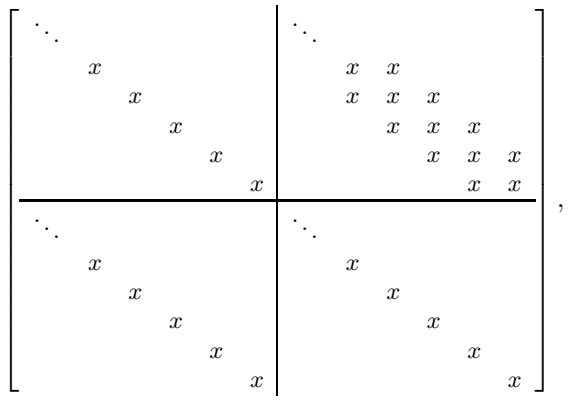
$$\left[\begin{array}{cccccc|cccccc}
\ddots & & & & & & \ddots & & & & & \\
 & x & & & & & & x & x & & & \\
 & & x & & & & & x & x & x & & \\
 & & & x & & & & & x & x & x & \\
 & & & & x & \oplus & & & & x & x & x \\
 & & & & \oplus & x & & & & & x & x \\
\hline
\ddots & & & & & & \ddots & & & & & \\
 & x & & & & & & x & & & & \\
 & & x & & & & & & x & & & \\
 & & & x & & & & & & x & & \\
 & & & & x & & & & & & x & \oplus \\
 & & & & & x & & & & & \oplus & x
\end{array}\right],$$

- a transformation $\widetilde{G}_n$ is not needed

- a symplectic Gauss transformation L_n to eliminate the entry $(n, n-1)$



- a transformation G_{n+1} is not needed
- a symplectic Givens transformation G_n to eliminate the entry $(2n, 2n-1)$



there is no b_1.

The bulge has been chased out of the matrix, the final two transformations are not needed here. The non-parameterized version of the algorithm is summarized in Table 5.3. A corresponding MATLAB programme called `param_sr_implicit_quadruple` working only on the parameters is given in Appendix B. The implicit quadruple SR step working on the parameters only requires $\mathcal{O}(n)$ flops. As the entire process works only on the parameters which determine the Hamiltonian matrix, the Hamiltonian structure is forced in every step of the algorithm.

6. Solving the 2×2 and 4×4 subproblems. We proceed with the SR iteration until the problem has completely decoupled into Hamiltonian J-Hessenberg subproblems of size 2×2 or 4×4; that is, at least every second ζ_j in $\widehat{H} = S^{-1}HS$ is neglectably small. In a final step we now have to transform each of these subproblems into a form from which the eigenvalues can be read off. In order to do so, each subproblem is transformed into Hamiltonian Schur form H_{Schur} by an orthogonal symplectic transformation

$$H_{Schur} = \begin{bmatrix} T & N \\ 0 & -T^T \end{bmatrix},$$

where N is symmetric and T is a block upper triangular matrix with 1×1 or 2×2 blocks on the diagonal. This has already been discussed in [38] for the case that

Algorithm: Quadruple shift implicit SR step

Given a $2n \times 2n$ Hamiltonian matrix A in J-Hessenberg form compute a quadruple shift implicit SR step. That is, given either a complex shift $\gamma \in \mathbb{C}$, $\mathrm{Re}(\gamma) \neq 0$ or two shifts μ and η (either real or purely imaginary), the symplectic matrix S of the SR decomposition $(A-\mu I)(A+\mu I)(A-\eta I)(A+\eta I)$ (in case the shift $\gamma \in \mathbb{C}$ is given, $\mu = \gamma$, $\eta = \overline{\gamma}$) is computed implicitly. A will be overwritten by its J–Hessenberg form.

$x_1 = (\delta_1^2 + \nu_1\beta_1 - \eta^2)(\delta_1^2 + \nu_1\beta_1 - \mu^2) + \nu_1\nu_2\zeta_2^2$
$x_2 = \nu_1\zeta_2[(\delta_1^2 + \nu_1\beta_1 - \eta^2) + \delta_2^2 + \nu_2\beta_2 - \mu^2]$
$x_3 = \nu_1\nu_2\zeta_2\zeta_3$
compute $\widetilde{G}_2$ such that $\widetilde{G}_2(x_2e_2 + x_3e_3) = \alpha_1 e_2$
$A = \widetilde{G}_2 A \widetilde{G}_2^T$
$S = \widetilde{G_2}$
compute $\widetilde{G}_1$ such that $\widetilde{G}_1(x_1e_1 + \alpha_1 e_2) = \alpha_2 e_1$
$A = \widetilde{G}_1 A \widetilde{G}_1^T$
$S = \widetilde{G_1} S$
for $j = 1 : n-1$
  for $k = \min(n, j+2) : -1 : j+1$
    compute G_k such that $(G_k A)_{k+n,j} = 0$
    $A = G_k A G_k^T$
  end
  if $j < n-1$
  then compute $\widetilde{G}_{j+1}$ such that $(\widetilde{G}_{j+1}A)_{j+2,j} = 0$
    $A = \widetilde{G}_{j+1} A \widetilde{G}_{j+1}^T$
  end
  if $A_{j+1,j} \neq 0$ and $A_{n+j,n+j} = 0$
  then stop, reduction does not exist
  end
  compute L_{j+1} such that $(L_{j+1}A)_{j+1,j} = 0$
  $A = L_{j+1} A L_{j+1}^{-1}$
  for $k = \min(n, j+2) : -1 : j+1$
    compute G_k such that $(G_k A)_{n+k,n+j} = 0$
    $A = G_k A G_k^T$
  end
  if $j < n-1$
  then compute H_j such that $(H_j A)_{j+2:\min(j+3,n),n+j} = 0$
    $A = H_j A H_j^T$
  end
end

TABLE 5.3
Quadruple shift implicit SR step

the Hamiltonian matrix has no purely imaginary eigenvalues. In that paper, the SR algorithm is used to solve complete stable eigenproblems, this implies that the 2×2 and 4×4 subproblems have no purely imaginary eigenvalues. This cannot be assumed

here. Even if the original Hamiltonian matrix H does not have purely imaginary eigenvalues, the projected much smaller Hamiltonian J-Hessenberg matrix computed by the symplectic Lanczos method for Hamiltonian matrices may have some. When possible, the results from [38] are used. The cases not considered in [38] are discussed here in detail.

Let us start with the 2×2 subproblems. They are of the form

$$H_{2\times 2} = \begin{bmatrix} \delta_j & \beta_j \\ \nu_j & -\delta_j \end{bmatrix}.$$

The characteristic polynomial is given by

$$\det(H_{2\times 2} - \lambda I) = \lambda^2 - (\delta_j^2 + \beta_j \nu_j).$$

Hence, the eigenvalues are

$$\pm\lambda = \pm\sqrt{\delta_j^2 + \beta_j \nu_j}.$$

In case the eigenvalues are real that is, if $\delta_j^2 + \beta_j \nu_j > 0$, there are two options to transform $H_{2\times 2}$ into upper triangular form. Either the positive eigenvalue λ is put into the $(1,1)$ position

$$\begin{bmatrix} \lambda & x \\ & -\lambda \end{bmatrix},$$

or the negative one

$$\begin{bmatrix} -\lambda & x \\ & \lambda \end{bmatrix}.$$

As for most applications the stable invariant subspace is sought, we choose the latter option. Please note, that we do not assume that the Hamiltonian eigenproblem to be solved has no purely imaginary eigenvalues. Hence, a stable invariant subspace might not exist. In order to put the negative eigenvalue in the $(1,1)$ position, compute a 2×2 Givens transformation

$$Q = \begin{cases} \sqrt{(\delta_j - \lambda)^2 + \nu_j^2}^{-1} \begin{bmatrix} \delta_j - \lambda & -\nu_j \\ \nu_j & \delta_j - \lambda \end{bmatrix} & \text{if } \lambda \neq \delta_j \quad \text{and} \quad \nu_j \neq 0, \\ I & \text{if } \lambda = -\delta_j \quad \text{and} \quad \nu_j = 0, \\ \sqrt{(4\delta_j^2 + \beta_j^2}^{-1} \begin{bmatrix} \beta_j & -2\delta_j \\ 2\delta_j & \beta_j \end{bmatrix} & \text{if } \lambda = \delta_j \quad \text{and} \quad \nu_j = 0. \end{cases}$$

Q is orthogonal and symplectic and

$$Q^T H_{2\times 2} Q = \begin{bmatrix} -\lambda & x \\ & \lambda \end{bmatrix}.$$

Lifting Q into the $(j, n+j)$th plane of a corresponding $2n \times 2n$ orthogonal symplectic Givens matrix G and transforming $\widehat{H}$ with G will eliminate ν_j and put $-\lambda$ in the (j,j) position of $G^T \widehat{H} G$. Therefore, SG contains in its jth column the eigenvector corresponding to $-\lambda$ which belongs to the stable invariant subspace (if there exists one).

REMARK 6.1. *In [145], it is suggested to further reduce the* 2×2 *problem by using the symplectic transformation*

$$Q' = \begin{bmatrix} \frac{1}{2\lambda} & x \\ & 2\lambda \end{bmatrix},$$

which transforms $Q^T H_{2\times 2} Q$ *into diagonal form*

$$Q'^{-1} Q^T H_{2\times 2} Q Q' = \begin{bmatrix} -\lambda & \\ & \lambda \end{bmatrix}.$$

This allows to read off the eigenvectors to both eigenvalues, but, depending on λ *this additional transformation may increase the numerical instability of the SR algorithm.*

This additional transformation is not necessary as both eigenvectors can be read off directly from

$$\begin{bmatrix} -\lambda & x \\ & \lambda \end{bmatrix}.$$

The eigenvector corresponding to $-\lambda$ *is* $e_1 = [1,\ 0]^T$*, the eigenvector corresponding to* λ *is*

$$\begin{bmatrix} x \\ 2\lambda \end{bmatrix}.$$

In case the eigenvalues are purely imaginary (that is, if $\delta_j^2 + \beta_j \nu_j < 0$) nothing needs to be done. If one is interested in at least one eigenvector corresponding to the pair of imaginary eigenvalues, then $H_{2\times 2}$ has to be transformed into its canonical Schur form

$$H_{final} = \begin{bmatrix} 0 & \pm\beta \\ \mp\beta & 0 \end{bmatrix}.$$

From

$$H[x\ y] = [x\ y] H_{final}, \qquad x, y \in \mathbb{R}^{2n}$$

we obtain with $z = x + iy$ for the case $Hx = -\beta y, Hy = \beta x$

$$Hz = i\beta z,$$

and just as for the case $Hx = \beta y, Hy = -\beta x$

$$Hz = -i\beta z.$$

Hence, an eigenvector can be read off; the eigenvector to the other eigenvalue is $\overline{z}$.

There exists an orthogonal Givens transformation such that

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \delta_j & \beta_j \\ \nu_j & -\delta_j \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} 0 & d \\ b & 0 \end{bmatrix}, \tag{6.1}$$

where $b, d \in \mathbb{R}, bd < 0$ and $\lambda = \pm i\beta = \pm\sqrt{bd} = \pm\sqrt{\delta_j^2 + \beta_j \nu_j}$. Equating the $(1,1)$ and the $(2,2)$ entry of both sides of the equation, yields

$$(c^2 - s^2)\delta_j - cs(\beta_j + \nu_j) = 0.$$

Dividing by $c^2\delta_j$ and introducing the new unknown $t = \frac{s}{c} = \frac{\sin(\psi)}{\cos(\psi)} = \tan(\psi)$ and $\tau = (\beta_j + \nu_j)/(2\delta_j)$ we obtain

$$t^2 + 2\tau t - 1 = 0.$$

Choosing

$$t = \frac{\operatorname{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}, \quad c = \frac{1}{\sqrt{1+t^2}}, \quad s = ct$$

determines the transformation. For a numerical sound implementation of this step, see, e.g., the LAPACK routine lanev2f [4]. In our actual implementation, we use MATLAB's `schur` function for solving the 2×2 problem in case there are purely imaginary eigenvalues, as this will compute (6.1) right away.

If in (6.1), $b < 0$ and $d > 0$, then the transformation

$$\begin{bmatrix} 1/x & \\ & x \end{bmatrix} \begin{bmatrix} 0 & d \\ b & 0 \end{bmatrix} \begin{bmatrix} x & \\ & 1/x \end{bmatrix} = \begin{bmatrix} 0 & \beta \\ -\beta & 0 \end{bmatrix}, \quad x = \sqrt{-\beta/b} \tag{6.2}$$

puts (6.1) into its canonical Schur form. Note, that the transformation matrix is symplectic. If in (6.1), $b > 0$ and $d < 0$, then the transformation

$$\begin{bmatrix} 1/x & \\ & x \end{bmatrix} \begin{bmatrix} 0 & d \\ b & 0 \end{bmatrix} \begin{bmatrix} x & \\ & 1/x \end{bmatrix} = \begin{bmatrix} 0 & -\beta \\ \beta & 0 \end{bmatrix}, \quad x = \sqrt{\beta/b} \tag{6.3}$$

puts (6.1) into its canonical Schur form.

REMARK 6.2. *In case, a different ordering of the eigenvalues on the diagonal is desired, a reordering of the eigenvalues is possible later using the idea presented in [65, Chapter 7.6.2]. Suppose*

$$H' = \begin{bmatrix} \lambda_1 & h_{12} \\ 0 & \lambda_2 \end{bmatrix},$$

and that we wish to reverse the order of the eigenvalues. Note that $H'x = \lambda_2 x$ where

$$x = \begin{bmatrix} h_{12} \\ \lambda_2 - \lambda_1 \end{bmatrix}.$$

Let Q_D be a Givens rotation such that the second component of $Q_D^T x$ is zero. Then

$$Q_D^T H'(Q_D e_1) = \lambda_2 Q_D^T (Q_D e_1) = \lambda_2 e_1$$

and so $Q_D^T H' Q_D$ must be of the form

$$Q_D^T H' Q_D = \begin{bmatrix} \lambda_2 & \pm h_{12} \\ 0 & \lambda_1 \end{bmatrix}.$$

Now, let us turn to the 4×4 subproblems. They are of the form

$$H_{4\times 4} = \left[\begin{array}{cc|cc} \delta_j & 0 & \beta_j & \zeta_{j+1} \\ 0 & \delta_{j+1} & \zeta_{j+1} & \beta_{j+1} \\ \hline \nu_j & 0 & -\delta_j & 0 \\ 0 & \nu_{j+1} & 0 & -\delta_{j+1} \end{array}\right], \qquad \zeta_{j+1} \neq 0.$$

The characteristic polynomial is given by

$$\begin{aligned}\det(H_{4\times 4}-\lambda I) &= \lambda^4-(\delta_j^2+\delta_{j+1}^2+\beta_j\nu_j+\beta_{j+1}\nu_{j+1})\lambda^2\\ &\quad+\delta_j^2\delta_{j+1}^2+\delta_j^2\beta_{j+1}\nu_{j+1}+\delta_{j+1}^2\beta_j\nu_j+\nu_j\nu_{j+1}\beta_j\beta_{j+1}-\nu_j\nu_{j+1}\zeta_{j+1}^2\\ &= \lambda^4-(a_j+a_{j+1})\lambda^2+a_ja_{j+1}-\nu_j\nu_{j+1}\zeta_{j+1}^2,\end{aligned}$$

where we used as before

$$a_j=\delta_j^2+\nu_j\beta_j.$$

Hence, the eigenvalues are given as

$$\begin{aligned}\lambda_{1/2/3/4} &= \pm\sqrt{-\frac{a_j+a_{j+1}}{2}\pm\sqrt{\left(\frac{a_j+a_{j+1}}{2}\right)^2-a_ja_{j+1}+\nu_j\nu_{j+1}\zeta_{j+1}^2}}\\ &= \pm\sqrt{-\frac{a_j+a_{j+1}}{2}\pm\sqrt{\left(\frac{a_j-a_{j+1}}{2}\right)^2+\nu_j\nu_{j+1}\zeta_{j+1}^2}}.\end{aligned}$$

In case the term under the inner square root is nonnegative,

$$\theta_j:=\left(\frac{a_j-a_{j+1}}{2}\right)^2+\nu_j\nu_{j+1}\zeta_{j+1}^2\geq 0,$$

there will be just real or purely imaginary eigenvalues and only if θ_j is negative, there will be complex eigenvalues with nonnegative real part. Obviously, in case ν_j or ν_{j+1} is zero (or both of them), the eigenvalues will be real or purely imaginary, but never complex with nonzero real part. As this is easily detected by inspecting the matrix entries, we will deal with both cases separately (in contrast to [38]).

First, let us assume that $\theta_j<0$. This implies that $\nu_j\nu_{j+1}\neq 0$. Let λ and $\mu=\overline{\lambda}$ be the eigenvalues with positive real part. Then

$$\begin{aligned}H'_{4\times 4} &= (H_{4\times 4}-\lambda I)(H_{4\times 4}-\mu I)\\ &= \left[\begin{array}{cc|cc} (\delta_j-\lambda)(\delta_j-\mu)+\beta_j\nu_j & \nu_{j+1}\zeta_{j+1} & x & x\\ \nu_j\zeta_{j+1} & (\delta_{j+1}-\lambda)(\delta_{j+1}-\mu)+\beta_{j+1}\nu_{j+1} & x & x\\ \hline -(\lambda+\mu)\nu_j & 0 & x & x\\ 0 & -(\lambda+\mu)\nu_{j+1} & x & x\end{array}\right]\\ &= \left[\begin{array}{cc|cc} h_{11} & h_{12} & x & x\\ h_{21} & h_{22} & x & x\\ \hline h_{31} & 0 & x & x\\ 0 & h_{42} & x & x\end{array}\right]\end{aligned}$$

is a real matrix of rank 2 (here only the entries we are interested in are given explicitly). As $\nu_j\nu_{j+1}\neq 0$, the entries in positions $(3,1)$, $(2,1)$ and $(4,2)$ of $H'_{4\times 4}$ can be eliminated by premultiplying with an appropriate sequence of orthogonal symplectic transformations (a Givens transformation $G_1=G(1,c_1,s_1)$ followed by a Givens type II transformation $\widetilde{G}_1=\widetilde{G}(1,c_2,s_2)$ and a Givens transformation G_2, resp.), then the

result will be of the form

$$
\begin{aligned}
Q^T H'_{4\times 4} &= G_2^T \widetilde{G}_1^T G_1^T H'_{4\times 4} \\
&= G_2^T \widetilde{G}_1^T \left[\begin{array}{cc|cc} c_1h_{11}-s_1h_{31} & c_1h_{12} & x & x \\ h_{21} & h_{22} & x & x \\ \hline 0 & s_1h_{12} & x & x \\ 0 & h_{42} & x & x \end{array}\right], \qquad c_1h_{31}+s_1h_{11}=0, \\
&= G_2^T \left[\begin{array}{cc|cc} x & x & x & x \\ 0 & s_2c_1h_{12}+c_2h_{22} & x & x \\ \hline 0 & c_2s_1h_{12}-s_2h_{42} & x & x \\ 0 & s_2s_1h_{12}+c_2h_{42} & x & x \end{array}\right], \qquad c_2h_{21}+s_2(c_1h_{11}-s_1h_{31})=0,
\end{aligned}
$$

where

$$
\begin{aligned}
c_2s_1h_{12}-s_2h_{42} &= -\frac{\sqrt{h_{11}^2+h_{31}^2}}{\sqrt{y}}\frac{h_{31}}{\sqrt{h_{11}^2+h_{31}^2}}h_{12}+\frac{h_{21}}{\sqrt{y}}h_{42} \\
&= \frac{h_{21}h_{42}-h_{31}h_{12}}{\sqrt{y}} \\
&= 0,
\end{aligned}
$$

as

$$
\begin{aligned}
c_1 &= h_{11}/\sqrt{h_{11}^2+h_{31}^2}, \\
s_1 &= -h_{31}/\sqrt{h_{11}^2+h_{31}^2}, \\
c_2 &= (c_1h_{11}-s_1h_{31})/\sqrt{h_{21}^2+(c_1h_{11}-s_1h_{31})^2} = \sqrt{h_{11}^2+h_{31}^2}/\sqrt{y} \\
s_2 &= -h_{21}/\sqrt{y}.
\end{aligned}
$$

That is, the $(3,2)$ element of $\widetilde{G}_1^T G_1^T H'_{4\times 4}$ is zero. As $H'_{\times 4}$ is of rank 2, this further implies that the $(3,3)$ and the $(3,4)$ element have to be zero as well. Hence,

$$
\begin{aligned}
Q^T H'_{4\times 4} &= G_2^T \left[\begin{array}{cc|cc} x & x & x & x \\ 0 & x & x & x \\ \hline 0 & 0 & 0 & 0 \\ 0 & x & x & x \end{array}\right] \\
&= \left[\begin{array}{cc|cc} x & x & x & x \\ 0 & x & x & x \\ \hline 0 & 0 & x & x \\ 0 & 0 & x & x \end{array}\right],
\end{aligned}
$$

where Q is the product of the orthogonal symplectic transformations used. The zero in position $(3,2)$ occurs after eliminating the $(3,1)$ and the $(2,1)$ entry due to the special form of $H'_{4\times 4}$ and it is not affected by the final transformation. Furthermore, it is obvious that the $(1,1)$ and the $(2,2)$ entry are nonzero as ν_j and ν_{j+1} are nonzero. Therefore, the rank condition assures that

$$
Q^T H'_{4\times 4} = \left[\begin{array}{cc|cc} x & x & x & x \\ 0 & x & x & x \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right].
$$

Thus, the last two rows of Q^T span the left-invariant subspace of $H'_{4\times 4}$ corresponding to λ and μ. Therefore, we have

$$Q^T H_{4\times 4} Q = \left[\begin{array}{cc|cc} x_1 & x_2 & x & x \\ x_3 & x_4 & x & x \\ \hline 0 & 0 & -x_1 & -x_3 \\ 0 & 0 & -x_2 & -x_4 \end{array}\right] = \left[\begin{array}{cc} \Delta & X \\ 0 & -\Delta^T \end{array}\right], \tag{6.4}$$

where the 2×2 matrix Δ has the eigenvalues $-\lambda, -\mu$ with negative real part. Using the same approach as in (6.1) an orthogonal matrix U_1 can be found such that

$$U_1^T \left[\begin{array}{cc} x_1 & x_2 \\ x_3 & x_4 \end{array}\right] U_1 = \left[\begin{array}{cc} -\alpha & d \\ b & -\alpha \end{array}\right],$$

where $\alpha, b, d \in \mathbb{R}$, and $-\lambda = -\alpha + i\beta, -\mu = -\alpha - i\beta$ with $i\beta = \sqrt{bd}$. Using

$$U = \operatorname{diag}(U_1, U_1),$$

yields

$$U^T Q^T H_{4\times 4} Q U = \left[\begin{array}{cc|cc} -\alpha & d & x & x \\ b & -\alpha & x & x \\ \hline 0 & 0 & \alpha & -b \\ 0 & 0 & -d & \alpha \end{array}\right].$$

In case eigenvectors have to be read off, transformations as in (6.2) or (6.3) have to be applied to achieve

$$\left[\begin{array}{cc|cc} -\alpha & \beta & x & x \\ -\beta & -\alpha & x & x \\ \hline 0 & 0 & \alpha & -\beta \\ 0 & 0 & \beta & \alpha \end{array}\right].$$

If we again lift QU into the suitable $2n \times 2n$ orthogonal symplectic G, then SG has columns j and $j+1$ belonging to the (stable) invariant subspace of H. This completes the discussion for the case that the 4×4 subproblem has complex eigenvalues with nonzero real part.

REMARK 6.3. *In case, a different ordering of the diagonal blocks in (6.4) is desired, a reordering is possible later using the ideas presented in, e.g, [76, Section 4.5.4]. An orthogonal symplectic matrix U can be found such that*

$$U^T \left[\begin{array}{cc} \Delta & X \\ 0 & -\Delta^T \end{array}\right] U = \left[\begin{array}{cc} \widetilde{\Delta} & X \\ 0 & -\widetilde{\Delta}^T \end{array}\right] \tag{6.5}$$

where Δ has eigenvalue with negative real part and $\widetilde{\Delta}$ has eigenvalues with positive real part. If Y is the solution of the Lyapunov equation

$$\Delta Y - Y\Delta^T = X,$$

then Y is symmetric and consequently the columns of $[-Y, I]^T$ span an isotropic subspace of the 4×4 Hamiltonian matrix. Thus, there exists a symplectic QR decomposition

$$\left[\begin{array}{c} -Y \\ I \end{array}\right] = U \left[\begin{array}{c} R \\ 0 \end{array}\right].$$

By direct computation, it can be seen that U is an orthogonal symplectic matrix which produces a reordering of the form (6.5). In our implementation, the HAPACK *[23] routine haschord is called to perform this computation.*

Next let us consider the case that $\theta_j \geq 0$, that is case that the eigenvalues of $H_{4\times 4}$ are real or purely imaginary. In case $\nu_j\nu_{j+1} \neq 0$, we suggest to use a few implicit double shift SR steps in order to decouple the problem further into two 2×2 subproblems. These can be solved as discussed at the beginning of this section. The same approach is used in the case $\nu_j\nu_{j+1} = 0$ and either $\nu_j = 0$ or $\nu_{j+1} = 0$, but not both of them. If $\nu_{j+1} \neq 0$, $H_{4\times 4}$ should first be permuted using

$$J_P = \left[\begin{array}{cc|cc} & 1 & & \\ -1 & & & \\ \hline & & & 1 \\ & & -1 & \end{array}\right] \tag{6.6}$$

such that we have

$$H_{4\times 4} = \left[\begin{array}{cc|cc} \delta_j & 0 & \beta_j & \zeta_{j+1} \\ 0 & \delta_{j+1} & \zeta_{j+1} & \beta_{j+1} \\ \hline \nu_j & 0 & -\delta_j & 0 \\ 0 & 0 & 0 & -\delta_{j+1} \end{array}\right]$$

in both cases.

If $\nu_j = \nu_{j+1} = 0$ we have

$$H_{4\times 4} = \left[\begin{array}{cc|cc} \delta_j & 0 & \beta_j & \zeta_{j+1} \\ 0 & \delta_{j+1} & \zeta_{j+1} & \beta_{j+1} \\ \hline 0 & 0 & -\delta_j & 0 \\ 0 & 0 & 0 & -\delta_{j+1} \end{array}\right],$$

and the eigenproblem decouples right away. In case the eigenvalues appear in the wrong order on the diagonal, they can be reordered as follows. In order to interchange δ_{j+1} and $-\delta_{j+1}$ a rotation of the form

$$G = \left[\begin{array}{cccc} 1 & & & \\ & c & & s \\ & & 1 & \\ & -s & & c \end{array}\right], \qquad \begin{aligned} c &= \beta_{j+1}/\sqrt{\beta_{j+1} + 4\delta_{j+1}^2} \\ s &= 2\delta_{j+1}/\sqrt{\beta_{j+1} + 4\delta_{j+1}^2} \end{aligned}$$

needs to be applied

$$G^T H_{4\times 4} G = \left[\begin{array}{cc|cc} \delta_j & -s\zeta_{j+1} & \beta_j & c\zeta_{j+1} \\ 0 & -\delta_{j+1} & c\zeta_{j+1} & \beta_{j+1} \\ \hline 0 & 0 & -\delta_j & 0 \\ 0 & 0 & s\zeta_{j+1} & \delta_{j+1} \end{array}\right].$$

Similarly, δ_j and $-\delta_j$ can be interchanged;

$$G = \left[\begin{array}{cccc} c & & s & \\ & 1 & & \\ -s & & c & \\ & & & 1 \end{array}\right], \qquad \begin{aligned} c &= \beta_j/\sqrt{\beta_j + 4\delta_j^2} \\ s &= 2\delta_j/\sqrt{\beta_j + 4\delta_j^2} \end{aligned}$$

needs to be applied

$$G^T H_{4\times 4} G = \left[\begin{array}{cc|cc} -\delta_j & 0 & \beta_j & c\zeta_{j+1} \\ -s\zeta_{j+1} & \delta_{j+1} & c\zeta_{j+1} & \beta_{j+1} \\ \hline 0 & 0 & \delta_j & s\zeta_{j+1} \\ 0 & 0 & 0 & -\delta_{j+1} \end{array}\right].$$

An invariant subspace can be read off from this form. In case, eigenvectors are desired, a similarity transformation with J_P as in (6.6) will achieve

$$\left[\begin{array}{cc|cc} \delta_{j+1} & s\zeta_{j+1} & \beta_{j+1} & -c\zeta_{j+1} \\ 0 & -\delta_j & -c\zeta_{j+1} & \beta_j \\ \hline 0 & 0 & -\delta_{j+1} & 0 \\ 0 & 0 & -s\zeta_{j+1} & \delta_j \end{array}\right].$$

In case both eigenvalues have to be interchanged, that is, the $(1,1)$ and the $(2,2)$ block have to be interchanged, this can be done as described in Remark 6.3.

REMARK 6.4. *In [145], it is suggested to further reduce the* 4×4 *problem*

$$H'_{4\times 4} = \left[\begin{array}{cc} \Delta_j & X_j \\ 0 & -\Delta_j^T \end{array}\right],$$

where Δ_j *contains the stable eigenvalues* $-\lambda, -\overline{\lambda}$*, to block diagonal form. The column range of* $U_j = (H'_{4\times 4} + \lambda_j I)(H'_{4\times 4} + \overline{\lambda} I)$ *spans the unstable invariant subspace. This implies*

$$\mathrm{span}(U_j) = \mathrm{span}(\left[\begin{array}{c} \Delta_j X_j - X_j \Delta_j^T + 2\mathrm{Re}(\lambda)\Delta_j \\ -4\mathrm{Re}(\lambda)\Delta_j^T \end{array}\right]) = \mathrm{span}(\left[\begin{array}{c} Z_1 \\ Z_2 \end{array}\right]).$$

As Z_2 *is regular, we can define*

$$F_j = \left[\begin{array}{cc} Z_2^{-T} & Z_1 \\ 0 & Z_2 \end{array}\right].$$

F_j *is symplectic and*

$$F_j^{-1} H'_{4\times 4} F_j = \left[\begin{array}{cc} \Delta_j & \\ & -\Delta_j^T \end{array}\right].$$

If Δ_j *is further transformed to diagonal form (using complex arithmetic), this allows to read off the eigenvectors to all eigenvalues, but, depending on* λ *this additional transformation may increase the numerical instability of the SR algorithm.*

Such an additional transformation is not necessary as the eigenvectors can be read off directly from

$$\left[\begin{array}{cc} \Delta & X \\ 0 & -\Delta^T \end{array}\right], \quad \textit{with} \quad \Delta = \left[\begin{array}{cc} -\alpha & \beta \\ -\beta & -\alpha \end{array}\right], \alpha, \beta \in \mathbb{R}, \alpha > 0.$$

The vectors

$$z_1 = \left[\begin{array}{c} 1 \\ i \\ 0 \\ 0 \end{array}\right], \quad z_2 = \left[\begin{array}{c} i \\ 1 \\ 0 \\ 0 \end{array}\right],$$

are eigenvectors corresponding to the eigenvalues

$$\lambda_1 = -\alpha + i\beta, \lambda_2 = -\alpha - i\beta.$$

In order to obtain the eigenvectors corresponding to the eigenvalues

$$\lambda_3 = \alpha + i\beta, \lambda_4 = \alpha - i\beta$$

consider for $j = 3, 4$

$$\begin{bmatrix} \Delta & X \\ 0 & -\Delta^T \end{bmatrix} z_j = \lambda_j z_j, \qquad z_j = \begin{bmatrix} z_{j1} \\ z_{j2} \end{bmatrix}, z_{j1}, z_{j2} \in \mathbb{C}^2.$$

This implies

$$-\Delta^T z_{j2} = \lambda_j z_{j2}, \quad \textit{and} \quad \Delta z_{j1} + X z_{j2} = \lambda_j z_{j1}.$$

The first system is solved easily

$$z_{32} = \begin{bmatrix} 1 \\ i \end{bmatrix}, \quad \textit{and} \quad z_{42} = \begin{bmatrix} i \\ 1 \end{bmatrix}.$$

This allows to rewrite the second system as

$$(\Delta - \lambda_j I) z_{j1} = -X z_{j2}.$$

As λ_j *is not an eigenvalue of* Δ*, this* 2×2 *systems of linear equations can easily be solved. This determines the eigenvectors*

$$z_3 = \begin{bmatrix} z_{31} \\ 1 \\ i \end{bmatrix}, \quad z_4 = \begin{bmatrix} z_{41} \\ i \\ 1 \end{bmatrix}.$$

When all 2×2 and 4×4 subproblems are solved, H has been transformed into a matrix of the form

$$\left[\begin{array}{cccc|cccc} H_{11} & & & & H_{1,m+1} & & & \\ & H_{22} & & & & H_{2,m+2} & & \\ & & \ddots & & & & \ddots & \\ & & & H_{mm} & & & & H_{m,2m} \\ \hline H_{m+1,1} & & & & -H_{11}^T & & & \\ & H_{m+2,2} & & & & -H_{22}^T & & \\ & & \ddots & & & & \ddots & \\ & & & H_{2m,m} & & & & -H_{mm}^T \end{array}\right]$$

where the blocks H_{ij} are either 1×1 or 2×2. If the block H_{jj} is of order 2×2, then the block $H_{m+j,j} = 0$ and H_{jj} has a pair of complex conjugate eigenvalues with negative real part. If the block H_{jj} is of order 1×1 and $H_{m+j,j} = 0$, then H_{jj} is a negative real eigenvalue of H, otherwise (that is, if $H_{m+j,j} \neq 0$) $H_{jj} = 0$ and the 2×2 submatrix

$$\begin{bmatrix} H_{jj} & H_{j,m+j} \\ H_{m+j,j} & -H_{jj}^T \end{bmatrix} = \begin{bmatrix} 0 & H_{j,m+j} \\ H_{m+j,j} & 0 \end{bmatrix}$$

represents a pair of purely imaginary eigenvalue.

Any order of the eigenvalues on the diagonal is possible. As

$$\begin{bmatrix} P^T & 0 \\ 0 & P^T \end{bmatrix} \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \begin{bmatrix} P & 0 \\ 0 & P \end{bmatrix} = \begin{bmatrix} P^TAP & P^TGP \\ P^TQP & -P^TAP \end{bmatrix},$$

we can easily rearrange the order of the eigenvalues by appropriate permutations. If, e.g., one wants to move the blocks corresponding to purely imaginary eigenvalues all the way to the end of the matrix, then this can be done as follows: Assume the purely imaginary eigenvalue is represented by the entries $H_{jj}, H_{m+j,j}, H_{j,m+j}$ and $H_{j+1,j+1}$ is a 2×2 block, then a permutation

$$\begin{bmatrix} 0 & I_2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} H_{xy} & 0 \\ 0 & H_{x+1,y+1} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ I_2 & 0 \end{bmatrix} = \begin{bmatrix} H_{x+1,y+1} & 0 \\ 0 & H_{xy} \end{bmatrix}, x, y = j, m+j$$

will interchange the blocks H_{jj} and $H_{j+1,j+1}$, as well as the corresponding blocks $H_{j,m+j}$ and $H_{j+1,m+j+1}$, $H_{m+j,j}$ and $H_{m+j+1,j+1}$, and $H_{m+j,m+j}$ and $H_{m+j+1,m+j+1}$. If $H_{j+1,j+1}$ is a 1×1 block, then a permutation with

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

will interchange the blocks. Proceeding in this way, all purely imaginary eigenvalue can be moved to the end of the matrix such that

$$\left[\begin{array}{ccccccc|ccccccc} H_{11} & & & & & & & H_{1,r} & & & & & & \\ & \ddots & & & & & & & \ddots & & & & & \\ & & H_{kk} & & & & & & & H_{k,\ell} & & & & \\ & & & 0 & & & & & & & H_{q,p} & & & \\ & & & & \ddots & & & & & & & \ddots & & \\ & & & & & 0 & & & & & & & H_{m,2m} \\ \hline 0 & & & & & & & -H_{11}^T & & & & & & \\ & \ddots & & & & & & & \ddots & & & & & \\ & & 0 & & & & & & & -H_{kk}^T & & & & \\ & & & H_{p,q} & & & & & & & 0 & & & \\ & & & & \ddots & & & & & & & \ddots & & \\ & & & & & H_{2m,m} & & & & & & & 0 & \end{array}\right] \tag{6.7}$$

where $\ell = m + k, p = \ell + 1, q = k + 1, r = m + 1$ and the blocks H_{11} to H_{kk} represent the real or complex eigenvalues with negative real part. Clearly, in case the symplectic transformation matrix S required to transform H into this form has been accumulated, eigenvectors can be read off for all eigenvalues with nonnegative real part and for all purely imaginary ones, if, for all complex eigenvalues, the corresponding 2×2 matrix has been transformed to

$$\begin{bmatrix} -\alpha & \beta \\ -\beta & -\alpha \end{bmatrix}.$$

6.1. Eigenvectors. Eigenvectors can be read off the Schur-like form (6.7) obtained via the SR iteration. Let us first consider the case of real eigenvalues (see

Remark 6.1) and partition the $2n \times 2n$ matrix (6.7) as

$$\left[\begin{array}{ccccc|ccccc} \widetilde{H}_1 & & & & & \widetilde{H}_3 & & & & \\ & -\lambda & & & & & \widetilde{h} & & & \\ & & \widetilde{H}_2 & & & & & \widetilde{H}_4 & & \\ & & & 0 & & & & & \widetilde{H}_5 & \\ \hline 0 & & & & & -\widetilde{H}_1^T & & & & \\ & 0 & & & & & \lambda & & & \\ & & 0 & & & & & -\widetilde{H}_2^T & & \\ & & & -\widetilde{H}_5 & & & & & 0 & \end{array}\right],$$

with $\lambda, \widetilde{h} \in \mathbb{R}$ and $\widetilde{H}_1, \widetilde{H}_3 \in \mathbb{R}^{n_1 \times n_1}, \widetilde{H}_2, \widetilde{H}_4 \in \mathbb{R}^{n_2 \times n_2}, \widetilde{H}_5 \in \mathbb{R}^{n_3 \times n_3}$ where $n = n_1 + 1 + n_2 + n_3$. Obviously, the eigenvector corresponding to $-\lambda$, $\lambda \in \mathbb{R}, \lambda > 0$ can be read off. The eigenvector corresponding to λ

$$\left[\begin{array}{ccccc|ccccc} \widetilde{H}_1 & & & & & \widetilde{H}_3 & & & & \\ & -\lambda & & & & & \widetilde{h} & & & \\ & & \widetilde{H}_2 & & & & & \widetilde{H}_4 & & \\ & & & 0 & & & & & \widetilde{H}_5 & \\ \hline 0 & & & & & -\widetilde{H}_1^T & & & & \\ & 0 & & & & & \lambda & & & \\ & & 0 & & & & & -\widetilde{H}_2^T & & \\ & & & -\widetilde{H}_5 & & & & & 0 & \end{array}\right] \left[\begin{array}{c} 0 \\ \widetilde{h} \\ 0 \\ 0 \\ 0 \\ 2\lambda \\ 0 \\ 0 \end{array}\right] = \lambda \left[\begin{array}{c} 0 \\ \widetilde{h} \\ 0 \\ 0 \\ 0 \\ 2\lambda \\ 0 \\ 0 \end{array}\right].$$

Next let us consider complex eigenvalues with nonzero real part (see Remark 6.4). Partition the $2n \times 2n$ matrix (6.7) as

$$\left[\begin{array}{ccccc|ccccc} \widetilde{H}_1 & & & & & \widetilde{H}_3 & & & & \\ & \Delta & & & & & \widetilde{h} & & & \\ & & \widetilde{H}_2 & & & & & \widetilde{H}_4 & & \\ & & & 0 & & & & & \widetilde{H}_5 & \\ \hline 0 & & & & & -\widetilde{H}_1^T & & & & \\ & 0 & & & & & -\Delta^T & & & \\ & & 0 & & & & & -\widetilde{H}_2^T & & \\ & & & -\widetilde{H}_5 & & & & & 0 & \end{array}\right], \tag{6.8}$$

with $\Delta, \widetilde{h} \in \mathbb{R}^{2\times 2}$ and $\widetilde{H}_1, \widetilde{H}_3 \in \mathbb{R}^{n_1 \times n_1}, \widetilde{H}_2, \widetilde{H}_4 \in \mathbb{R}^{n_2 \times n_2}, \widetilde{H}_5 \in \mathbb{R}^{n_3 \times n_3}$ where $n = n_1 + 2 + n_2 + n_3$. Moreover,

$$\Delta = \left[\begin{array}{cc} -\alpha & \beta \\ -\beta & -\alpha \end{array}\right], \qquad \alpha, \beta \in \mathbb{R}, \alpha > 0.$$

The eigenvectors z_1 and z_2 corresponding to the eigenvalues of Δ, that is, corresponding to $\lambda_1 = -\alpha + i\beta$ and $\lambda_2 = -\alpha - i\beta$ are given as (using the same partition as in

the matrix (6.8))

$$z_1 = \left[\begin{array}{c} 0 \\ z_{12} \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right], z_{12} = \left[\begin{array}{c} 1 \\ i \end{array}\right] \qquad z_2 = \left[\begin{array}{c} 0 \\ z_{22} \\ 0 \\ 0 \\ \hline 0 \\ 0 \\ 0 \\ 0 \end{array}\right], z_{22} = \left[\begin{array}{c} i \\ 1 \end{array}\right].$$

The eigenvectors z_3 and z_4 corresponding to the eigenvalues of $-\Delta^T$, that is, corresponding to $\lambda_3 = \alpha + i\beta$ and $\lambda_4 = \alpha - i\beta$ are given as (using the same partition as in the matrix (6.8))

$$z_3 = \left[\begin{array}{c} 0 \\ z_{31} \\ 0 \\ 0 \\ \hline 0 \\ z_{32} \\ 0 \\ 0 \end{array}\right], z_{32} = \left[\begin{array}{c} 1 \\ i \end{array}\right] \qquad z_4 = \left[\begin{array}{c} 0 \\ z_{41} \\ 0 \\ 0 \\ \hline 0 \\ z_{42} \\ 0 \\ 0 \end{array}\right], z_{42} = \left[\begin{array}{c} i \\ 1 \end{array}\right]$$

where z_{31} and z_{41} are chosen as explained in Remark 6.4.

Finally, let us consider purely imaginary eigenvalues. Partition the $2n \times 2n$ matrix (6.7) as

$$\left[\begin{array}{cccc|cccc} \widetilde{H}_1 & & & & \widetilde{H}_2 & & & \\ & 0 & & & & \widetilde{H}_3 & & \\ & & 0 & & & & \beta & \\ & & & 0 & & & & \widetilde{H}_4 \\ \hline 0 & & & & -\widetilde{H}_1^T & & & \\ & -\widetilde{H}_3 & & & & 0 & & \\ & & -\beta & & & & 0 & \\ & & & -\widetilde{H}_4 & & & & 0 \end{array}\right], \tag{6.9}$$

with $\beta \in \mathbb{R}$ and $\widetilde{H}_1, \widetilde{H}_2 \in \mathbb{R}^{n_1 \times n_1}, \widetilde{H}_3 \in \mathbb{R}^{n_2 \times n_2}, \widetilde{H}_4 \in \mathbb{R}^{n_3 \times n_3}$ where $n = n_1 + n_2 + 1 + n_3$. The eigenvectors z_1 and z_2 corresponding to the eigenvalues $\lambda_1 = i\beta$ and $\lambda_2 = -i\beta$ are given as (using the same partition as in the matrix (6.9))

$$z_1 = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \\ i \\ 0 \end{array}\right], \qquad z_2 = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ \hline 0 \\ 0 \\ -i \\ 0 \end{array}\right].$$

The eigenvectors of the original Hamiltonian matrix H can be obtained from the eigenvectors z of the matrix $\widetilde{H}$ (6.7) if the transformation matrix S which transforms

Algorithm: Inverse Iteration

Given a matrix H, an eigenvalue approximation μ and a starting vector x_0, an eigenvector x is computed: $Hx = \mu x$.

$x_0 = \frac{1}{||x_0||} x_0$
$k = 1$
while not satisfied
    solve $(H - \mu I)x_k = x_{k-1}$
    $x_k = \frac{1}{||x_k||} x_k$
    $k = k + 1$
end

TABLE 6.1
Inverse Iteration

H into the form (6.7) is accumulated:

$$HSz = S\widetilde{H}z = \lambda Sz.$$

In case S is badly conditioned or has not been accumulated, inverse iteration might be more advisable in order to compute an eigenvector of the original Hamiltonian J-Hessenberg matrix. This has been first discussed in [132]. The presentation given here follows that one in [132].

As the eigenvalues of the Hamiltonian J-Hessenberg matrix H are known, one approach for computing additional eigenvectors is to use inverse iteration. Essentially, for this iteration, the inverse of $H - \mu I$ is repeatedly multiplied against a starting vector. A reasonable implementation of this process is as given in Table 6.1 (see, e.g., [65]). Solving the linear system $(H - \mu I)x_k = x_{k-1}$ is the most expensive part of this iteration. As H is of J-Hessenberg form, this can cheaply be achieved by an LU decomposition [65] of H.

Let us consider a 8×8 example $H - \mu I$

$$\left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline \nu_1 & & & & -\delta_1-\mu & & & \\ & \nu_2 & & & & -\delta_2-\mu & & \\ & & \nu_3 & & & & -\delta_3-\mu & \\ & & & \nu_4 & & & & -\delta_4-\mu \end{array}\right].$$

The first step of Gaussian elimination will zero all entries in the first column of this matrix below the $(1,1)$ entry. This can easily be achieved by premultiplication with

$$L_1 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline -\frac{\nu_1}{\delta_1-\mu} & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right]$$

which yields $U_1 = L_1(H - \mu I)$

$$U_1 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & \nu_2 & & & & -\delta_2-\mu & & \\ & & \nu_3 & & & & -\delta_3-\mu & \\ & & & \nu_4 & & & & -\delta_4-\mu \end{array}\right],$$

with $v_{11} = -\delta_1 - \mu + \beta_1\ell_1, v_{12} = \zeta_2\ell_1$ and $\ell_1 = -\nu_1/(\delta_1 - \mu)$. Next the entries below the $(2, 2)$ entry in the second column of $L_1(H - \mu I)$ have to be eliminated. This is achieved by

$$L_2 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline -\frac{\nu_1}{\delta_1-\mu} & & & & 1 & & & \\ & -\frac{\nu_2}{\delta_2-\mu} & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right]$$

which yields$U_2 = L_2(H - \mu I)$

$$U_2 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & & & & v_{21} & v_{22} & v_{23} & \\ & & \nu_3 & & & & -\delta_3-\mu & \\ & & & \nu_4 & & & & -\delta_4-\mu \end{array}\right],$$

with $v_{21} = \zeta_2\ell_2, v_{22} = -\delta_2 - \mu + \beta_2\ell_2, v_{23} = \zeta_3\ell_2$ and $\ell_2 = -\nu_2/(\delta_2 - \mu)$. Clearly, the next two columns can be transformed into the desired form in the same fashion. Applying

$$L_4 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline -\frac{\nu_1}{\delta_1-\mu} & & & & 1 & & & \\ & -\frac{\nu_2}{\delta_2-\mu} & & & & 1 & & \\ & & -\frac{\nu_3}{\delta_3-\mu} & & & & 1 & \\ & & & -\frac{\nu_4}{\delta_4-\mu} & & & & 1 \end{array}\right]$$

yields $U_4 = L_4(H - \mu I)$

$$U_4 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & & & & v_{21} & v_{22} & v_{23} & \\ & & & & & v_{32} & v_{33} & v_{34} \\ & & & & & & v_{43} & v_{44} \end{array}\right],$$

with $v_{32} = \zeta_3\ell_3, v_{33} = -\delta_3 - \mu + \beta_3\ell_3, v_{34} = \zeta_4\ell_3$ and $\ell_3 = -\nu_3/(\delta_3 - \mu)$, and $v_{43} = \zeta_4\ell_4, v_{44} = -\delta_4 - \mu + \beta_4\ell_4$ and $\ell_4 = -\nu_4/(\delta_4 - \mu)$. Next, the entries below the $(5, 5)$ entry in the fifth column have to be eliminated. This can be done by

$$L_5 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline -\frac{\nu_1}{\delta_1-\mu} & & & & 1 & & & \\ & -\frac{\nu_2}{\delta_2-\mu} & & & -\frac{v_{21}}{v_{11}} & 1 & & \\ & & -\frac{\nu_3}{\delta_3-\mu} & & & & 1 & \\ & & & -\frac{\nu_4}{\delta_4-\mu} & & & & 1 \end{array}\right]$$

yields $U_5 = L_5(H - \mu I)$

$$U_5 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & & & & & \widehat{v}_{22} & v_{23} & \\ & & & & & v_{32} & v_{33} & v_{34} \\ & & & & & & v_{43} & v_{44} \end{array}\right],$$

with $\widehat{v}_{22} = v_{22} + v_{12}\ell_5$ and $\ell_5 = -v_{21}/v_{11}$. Continuing in this fashion, finally we obtain

$$L_7 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline -\frac{\nu_1}{\delta_1-\mu} & & & & 1 & & & \\ & -\frac{\nu_2}{\delta_2-\mu} & & & -\frac{v_{21}}{v_{11}} & 1 & & \\ & & -\frac{\nu_3}{\delta_3-\mu} & & & -\frac{v_{32}}{\widehat{v}_{22}} & 1 & \\ & & & -\frac{\nu_4}{\delta_4-\mu} & & & -\frac{v_{43}}{\widehat{\widehat{v}}_{33}} & 1 \end{array}\right]$$

yields $U_7 = L_7(H - \mu I)$

$$U_7 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & & & & & \widehat{v}_{22} & v_{23} & \\ & & & & & & \widehat{v}_{33} & v_{34} \\ & & & & & & & \widehat{v}_{44} \end{array}\right],$$

with $\widehat{v}_{33} = v_{33} + v_{23}\ell_6, \ell_6 = -v_{32}/\widehat{v}_{22}$ and $\widehat{v}_{44} = v_{44} + v_{34}\ell_7, \ell_7 = -v_{43}/\widehat{v}_{33}$. From this it is easily seen that in general the LU decomposition of a Hamiltonian J-Hessenberg matrix $H = LU$ is of the form

$$L = \left[\begin{array}{cc} 1 & \\ & \ddots \\ \ddots & 1 \\ & 1 \end{array}\right], \qquad U = \left[\begin{array}{cc} \diagdown & /// \\ & /// \end{array}\right].$$

The LU decomposition can be computed without forming H, L or U explicitly. From the parameters which determine a Hamiltonian J-Hessenberg matrix H, the entries of L and U can be computed directly. Only the $2n-1$ relevant entries of L as well as the $5n-2$ relevant entries of U should be stored. Linear systems $Hx = LUx = b$ can now easily be solved using forward substitution $Lz = b$ and backward substitution $Ux = z$. An implementation of this process should make use of the extremely sparse structure of L and U.

Pivoting should be incorporated in the process described above in order to increase numerical stability. Let us consider pivoting during the first n steps of the LU factorization first, that is, pivoting while eliminating the entries in the $(2,1)$ block. For simplicity, let us consider the 8×8 example discussed above. Assume that in U_1 we have $\nu_2 < \delta_2 - \mu$, then the second and the sixth row of U_1 should be interchanged. This gives

$$\widetilde{U}_1 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \nu_2 & & & & -\delta_2-\mu & & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & \delta_2-\mu & & & v_{11} & v_{12} & & \\ & & \nu_3 & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & & \nu_4 & & & -\delta_3-\mu & \\ & & & & & & & -\delta_4-\mu \end{array}\right].$$

As before, next, the entries below the $(2,2)$ entry in the second column of $\widetilde{U}_1$ have to be eliminated. This is achieved by

$$\widetilde{L}_2 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline & & & & 1 & & & \\ & -\frac{\delta_2-\mu}{\nu_2} & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right].$$

which yields $\widetilde{U}_2 = \widetilde{L}_2\widetilde{U}_1$

$$\widetilde{U}_2 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \nu_2 & & & & -\delta_2-\mu & & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{11} & v_{12} & & \\ & & & & \zeta_2 & \widetilde{v}_{22} & \zeta_2 & \\ & & \nu_3 & & & & -\delta_3-\mu & \\ & & & \nu_4 & & & & -\delta_4-\mu \end{array}\right],$$

with $\widetilde{v}_{22} = \beta_2 + \delta_2^2/\nu_2$. The structure of L and U is the same as before, but the symmetry in the $(1,2)$ block is lost.

Finally, let us consider pivoting during the last n steps of the LU factorization, that is, pivoting while eliminating the subdiagonal entries in the $(2,2)$ block. For simplicity, let us consider the 8×8 example discussed above. Assume that in U_4 we have $v_{21} < v_{11}$, then the fifth and the sixth row of U_4 should be interchanged. This gives

$$\widetilde{U}_4 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & v_{21} & v_{22} & v_{23} & \\ & & & & v_{11} & v_{12} & & \\ & & & & & v_{32} & v_{33} & v_{34} \\ & & & & & & v_{43} & v_{44} \end{array}\right].$$

Next, the entries below the $(5,5)$ entry in the fifth column have to be eliminated. This can be done by

$$\widetilde{L}_5 = \left[\begin{array}{cccc|cccc} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ \hline & & & & 1 & & & \\ & & & & -\frac{v_{11}}{v_{21}} & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{array}\right]$$

yields $\widetilde{U}_5 = \widetilde{L}_5\widetilde{U}_4$

$$\widetilde{U}_5 = \left[\begin{array}{cccc|cccc} \delta_1-\mu & & & & \beta_1 & \zeta_2 & & \\ & \delta_2-\mu & & & \zeta_2 & \beta_2 & \zeta_3 & \\ & & \delta_3-\mu & & & \zeta_3 & \beta_3 & \zeta_4 \\ & & & \delta_4-\mu & & & \zeta_4 & \beta_4 \\ \hline & & & & \widetilde{v}_{11} & \widetilde{v}_{12} & \widetilde{v}_{13} & \\ & & & & & \widetilde{v}_{22} & \widetilde{v}_{23} & \\ & & & & & v_{32} & v_{33} & v_{34} \\ & & & & & & v_{43} & v_{44} \end{array}\right],$$

with $\widetilde{v}_{11} = v_{21}, \widetilde{v}_{12} = v_{22}, \widetilde{v}_{13} = v_{23}, \widetilde{v}_{22} = v_{12} - v_{11}v_{22}/v_{21}, \widetilde{v}_{23} = -v_{11}v_{23}/v_{21}$. Hence, pivoting in the $(2,2)$ block will increase the number of upper subdiagonals in that block by one. The final LU decomposition of $H - \mu I$ will be of the form

$$L = \left[\begin{array}{cccc|ccccc} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & \ddots & & & & & & \\ & & & 1 & & & & & \\ \hline l_1 & & & & 1 & & & & \\ & \ddots & & & l_{n+1} & \ddots & & & \\ & & \ddots & & & \ddots & \ddots & & \\ & & & l_n & & & l_{2n-1} & 1 & \end{array}\right] \tag{6.10}$$

and

$$U = \left[\begin{array}{cccc|ccccc} u_1 & & & & b_1 & \zeta_{2,o} & & & \\ & \ddots & & & \zeta_{2,u} & \ddots & \ddots & & \\ & & \ddots & & & \ddots & \ddots & \ddots & \\ & & & \ddots & & & \ddots & \ddots & \zeta_{n,o} \\ & & & u_n & & & & \zeta_{n,u} & b_n \\ \hline & & & & v_{11} & v_{12} & v_{13} & & \\ & & & & & \ddots & \ddots & \ddots & \\ & & & & & & \ddots & \ddots & v_{n-2,n} \\ & & & & & & & \ddots & v_{n-1,n} \\ & & & & & & & & v_{nn} \end{array}\right] \tag{6.11}$$

Using the notation used above, the algorithm for computing the LU factorization of a Hamiltonian J-Hessenberg matrix can be summarized as given in Table 6.2. The pivot vector p denotes the interchange permutations. This information is needed in order to be able to perform the forward substitution as in the standard Gaussian elimination (see, e.g., [65, Chapter 7.6]). For completeness, the forward substitution $Lz = b$ is given in Table 6.3 in order to demonstrate the use of the pivot vector in order to perform the necessary row interchanges.

A careful flop count reveals that the LU decomposition requires at most $7n$ multiplications and $3n$ additions for a $2n \times 2n$ Hamiltonian J-Hessenberg matrix. The forward substitution requires at most $4n$ multiplications and $4n$ additions, while the backward substitution requires at most $7n$ multiplications and $5n$ additions. Hence, making use of the special structure of the Hamiltonian J-Hessenberg matrices, the inverse iteration can be implemented in $\mathcal{O}(n)$ flops compared to $\mathcal{O}(n^2)$ flops in the general case. Usually (that is, assuming that the eigenvalue μ has been computed via the SR algorithm is explained in the previous sections), only one iteration is required to produce an adequate approximate eigenvector.

Algorithm: LU factorization of a Hamiltonian J-Hessenberg matrix

Given a $2n \times 2n$ Hamiltonian matrix A in J-Hessenberg form compute the LU factorization of $A - \mu I$ for a given real or complex μ. The resulting matrices L and U are of the form (6.10) and (6.11), the pivot vector p defines the interchange permutations.

$p = 1 : 2n$
for $j = 1 : n$
  $u_j = \delta_j - \mu$
  $v_{jj} = -\delta_j - \mu$
  $b_j = \beta_j$
  if $j > 1$
  then $\zeta_{j,o} = \zeta_j$
    $\zeta_{j,u} = \zeta_j$
  end
end
for $j = 1 : n$
  if $|\nu_j| > |\delta_j - \mu|$
  then $p_j = n + j$
    $u_j = \nu_j$
    $\nu_j = \delta_j - \mu$
    if $j > 1$ **then** $v_{j,j-1} = \zeta_j$ **end**
    $v_{jj} = \beta_j$
    if $j < n$ **then** $v_{j,j+1} = \zeta_{j+1}$ **end**
    $\zeta_{j,u} = 0$
    $b_j = -\delta_j - \mu$
    $\zeta_{j+1,o} = 0$
    $\ell_j = -\nu_j / u_j$
    $v_{jj} = v_{jj} + \ell_j \beta_j$
  else $\ell_j = -\nu_j / u_j$
    if $j > 1$ **then** $v_{j+1,j} = \ell_j \zeta_{j+1}$ **end**
    $v_{jj} = v_{jj} + \beta_j \ell_j$
    if $j < n$ **then** $v_{j,j+1} = \ell_j \zeta_{j+1}$ **end**
  end **end**
for $j = 1 : n - 1$
  if $|v_{j+1,j}| > |v_{jj}|$
  then $p_{n+j} = n + j + 1$
    $h = v_{j,j:j+2}$
    $v_{j,j:j+2} = v_{j+1,j:j+2}$
    $v_{j+1,j:j+2} = h$
  end
  $\ell_{n+j} = -v_{j+1,1} / v_{jj}$
  $v_{j+1,j+1} = v_{j+1,j+1} + \ell_{n+j} v_{j,j+1}$
  $v_{j+1,j+2} = v_{j+1,j+2} + \ell_{n+j} v_{j,j+2}$
end

TABLE 6.2
LU factorization of a Hamiltonian J-Hessenberg matrix with pivoting

Algorithm: Forward Substitution using the LU factorization from Table 6.2

Given the LU factorization of a $2n \times 2n$ Hamiltonian matrix A in J-Hessenberg form, the system $Lz = x$ is solved. The pivot vector p defines the interchange permutations.

for $j = 1 : 2n$
  $h = x_j$
  $x_j = x_{p_j}$
  $x_{p_j} = h$
  if $j < n+$
  then $z_j = x_j$
  end
  if $j == n + 1$
  then $z_j = l_{j-n} z_{j-n} + x_j$
  end
  if $j > n + 1$
  then $z_j = l_{j-n} z_{j-n} + l_{j-1} z_{j-1} + x_j$
  end
end

TABLE 6.3
Forward substitution

7. Numerical Experiments. The parameterized SR algorithm for Hamiltonian matrices as described in the previous sections has been implemented in MATLAB Release 2006a. Numerical tests were run on a Pentium M processor. The tolerance for declaring deflation was chosen as

$$|\zeta_j| \leq 2^{-52}(|\delta_{j-1}| + |\delta_j|).$$

A symplectic Gauss transformation was reject when its condition number was larger than 10^8. This never happened for random test cases.

First numerical tests to determine the convergence properties of the algorithm have been performed. 100 random Hamiltonian J-Hessenberg matrices of the size $2n \times 2n$, $n = 3, \ldots, 200$ were generated and the average number of iterations needed for computing all eigenvalues has been determined. Here each implicit quadruple shift SR step was counted as one iteration, no matter how small the problem has become due to deflation. Figure 7.1 displays the average number of iterations needed as well as the maximal and the minimal number of iterations needed within the test set of 100 matrices. This data shows that on average, 0.706 iterations per eigenvalue are required. When considering only smaller matrices ($n = 3, \ldots, 20$), on average only 0.67 iterations per eigenvalue are required. This is comparable to the number of iterations needed by the QR algorithm. Similar tests have been performed in [38]. There a different implementation (not parameterized) has been used, the code used only single precision. In [38], it was reported that 'The number of iterations needed to compute all eigenvalues of the $2n \times 2n$ Hamiltonian J-Hessenberg matrix was between $2n$ and $4n$. The average number of iterations for the computation of an eigenvalue was between 1 and 1.5.' In our tests with the nonparameterized version of the SR algorithm we also observed an increase in the number of iterations as compared to

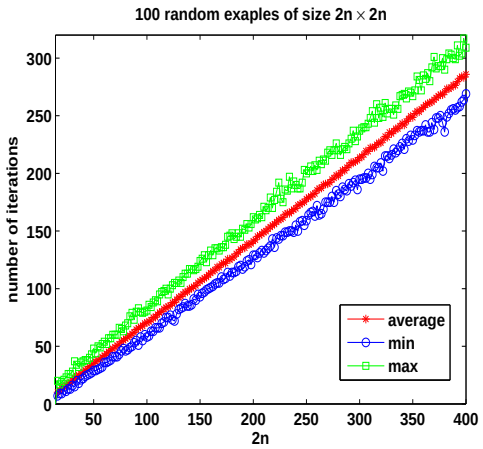


FIG. 7.1. *Average number of iterations needed to compute all eigenvalues*

the number of iterations needed by the parameterized version.

As the implementation works on the parameters only, each iteration step requires $\mathcal{O}(n)$ flops (not considering the update of the overall transformation matrix). Therefore, the overall work for the computation of the eigenvalues alone is $\mathcal{O}(n^2)$ plus $\mathcal{O}(n^3)$ for the initial reduction to J-Hessenberg form, which is comparable with the work for the QR algorithm for symmetric matrices.

In order to say something about the accuracy of the computed eigenvalues, let us consider the Hamiltonian J-Hessenberg matrix

$$H = \left[\begin{array}{cccccc|cccccc} 1 & & & & & & 19 & 2 & & & & \\ & 2 & & & & & 2 & 18 & 8 & & & \\ & & 3 & & & & & 8 & 17 & 5 & & \\ & & & 4 & & & & & 5 & 16 & 3 & \\ & & & & 5 & & & & & 3 & 15 & 6 \\ & & & & & 6 & & & & & 6 & 14 \\ \hline -3 & & & & & & -1 & & & & & \\ & -5 & & & & & & -2 & & & & \\ & & -7 & & & & & & -3 & & & \\ & & & -9 & & & & & & -4 & & \\ & & & & -11 & & & & & & -5 & \\ & & & & & -13 & & & & & & -6 \end{array}\right].$$

All eigenvalues are purely imaginary. The eigenvalues have been computed using the parameterized SR algorithm and the routine `haeig` from the HAPACK package, a different, structure-preserving eigensolvers for Hamiltonian eigenproblem. This routine does not make use of the Hamiltonian J-Hessenberg structure, but it is in contrast to the SR algorithm backward stable. Hence we expect that it will perform slightly better then the (parameterized) SR algorithm. For each computed eigenvalue λ_j, the minimal singular value σ_j^{min} of $H - \lambda_j I$ has been computed. In exact arithmetic, this value has to be zero.

σ^{min} `haeig`	σ^{min} SR	eigenpair
3.4843e-016	3.2871e-015	0 ± 6.1777e+000i
3.5513e-015	3.6666e-015	0 ± 7.5082e+000i
4.5359e-015	4.5359e-015	0 ± 8.1416e+000i
7.1138e-016	6.8828e-015	0 ± 1.0691e+001i
1.9564e-015	1.5470e-014	0 ± 1.3046e+001i
2.0029e-015	3.0869e-015	0 ± 1.3046e+001i

As expected, the eigenvalues computed by the SR algorithm are almost as accurate as those computed by `haeig`, only one digit might be lost.

Cubic convergence of the SR algorithm can clearly be seen for this example by considering the values ζ_j during the iterations:

iteration	ζ_4	ζ_5
0	5.0000e+000	3.0000e+000
1	-5.1783e+000	-6.4104e-001
2	6.6055e+000	5.8846e-003
3	-4.6184e+000	-6.5287e-009
4	1.6696e+000	1.4244e-022
5	9.2538e-003	0
6	1.3824e-009	0
7	-7.6066e-024	0

Finally, consider the following 4×4 problem

$$H = \left[\begin{array}{cc|cc} 3-\epsilon & 1 & -1 & -1 \\ 4 & 2-\epsilon & -1 & -1 \\ \hline 4\epsilon-11 & 2\epsilon-5 & -3+\epsilon & -4 \\ 2\epsilon-5 & 2\epsilon-2 & -1 & -2+\epsilon \end{array}\right] = \left[\begin{array}{cc} A & G \\ Q & -A^T \end{array}\right]$$

is taking from the collection of benchmark examples for the numerical solution of algebraic Riccati equations [24, Example 11]. It represents a type of algebraic Riccati equations arising in H_∞-control problems. The spectrum of H is $\{\pm\epsilon \pm \sqrt{-1}\}$; the eigenvalues approach the imaginary axis as $\epsilon \to 0$. The solution of the corresponding Riccati equation

$$0 = Q + A^T X + XA - XGX \tag{7.1}$$

can be computed using the stable invariant subspace of H; i.e., the subspace corresponding to the eigenvalues of H in the open left half plane. If this subspace is spanned by $\left[\begin{array}{c} U_1 \\ U_2 \end{array}\right]$ and U_1 is invertible, then $X = U_2 U_1^{-1}$ is the stabilizing solution of (7.1). For the example considered here, the matrix

$$X = \left[\begin{array}{cc} 2 & 1 \\ 1 & 1 \end{array}\right]$$

solves (7.1) for arbitrary ϵ. Computing the Hamiltonian J-Hessenberg form of H and using the direct approach of solving the resulting 4×4 problem as discussed in Section 6, we obtain

ϵ	$\|\|X - X_{SR}\|\|$	$\|\lambda - \lambda_{SR}\|$	$\|\lambda - \lambda_{haeig}\|$	$\|\lambda - \lambda_{eig}\|$
10^{-1}	$6.5 \cdot 10^{-15}$	$5.2 \cdot 10^{-15}$	$8.3 \cdot 10^{-15}$	$5.8 \cdot 10^{-15}$
10^{-2}	$3.1 \cdot 10^{-14}$	$2.9 \cdot 10^{-14}$	$1.1 \cdot 10^{-13}$	$6.3 \cdot 10^{-14}$
10^{-3}	$1.2 \cdot 10^{-12}$	$5.8 \cdot 10^{-13}$	$1.0 \cdot 10^{-12}$	$2.9 \cdot 10^{-13}$
10^{-4}	$5.6 \cdot 10^{-12}$	$5.6 \cdot 10^{-12}$	$5.4 \cdot 10^{-12}$	$3.9 \cdot 10^{-12}$
10^{-5}	$1.3 \cdot 10^{-10}$	$1.3 \cdot 10^{-10}$	$7.5 \cdot 10^{-11}$	$7.8 \cdot 10^{-12}$
10^{-6}	$9.8 \cdot 10^{-10}$	$9.8 \cdot 10^{-10}$	$8.1 \cdot 10^{-11}$	$2.4 \cdot 10^{-10}$
10^{-7}	$9.7 \cdot 10^{-9}$	$5.1 \cdot 10^{-9}$	$4.6 \cdot 10^{-9}$	$4.4 \cdot 10^{-9}$
10^{-8}	$4.4 \cdot 10^{-8}$	$6.7 \cdot 10^{-9}$	$2.4 \cdot 10^{-8}$	$3.2 \cdot 10^{-8}$
10^{-9}	$4.3 \cdot 10^{-7}$	$4.9 \cdot 10^{-10}$	$2.9 \cdot 10^{-8}$	$4.3 \cdot 10^{-8}$
10^{-10}	$6.1 \cdot 10^{-7}$	$2.5 \cdot 10^{-10}$	$3.4 \cdot 10^{-8}$	$3.7 \cdot 10^{-8}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$1.2 \cdot 10^{-7}$	$4.4 \cdot 10^{-9}$	$2.9 \cdot 10^{-8}$	$2.3 \cdot 10^{-8}$

where λ denotes the exact eigenvalue, λ_{SR} the eigenvalue computed via the above described process, λ_{haeig} the eigenvalue computed via the function `haeig` of the HAPACK library [23] and λ_{eig} the eigenvalue computed via MATLAB's `eig`. The accuracy of the computed eigenvalues degrades as ϵ is chosen smaller and smaller, but from $\epsilon = 10^7$ on, the error in the eigenvalues computed with the process described here is essentially of the order of 10^{-9}, no matter how small ϵ is chosen. The eigenvalues computed via the unstructured solver `eig` and those computed via the structure-preserving function `haeig` have slightly larger errors. As the structured eigensolvers compute the real and the imaginary part of the complex quadruple eigenvalue, all eigenvalues have the same error. The unstructured solver computes two pairs of complex conjugate eigenvalues, each with the same real and imaginary part. In this case, one might have two different errors when comparing the computed eigenvalues with the exact ones, only one of those errors is given above (the other one is of the same order of magnitude). Similarly, the accuracy of the computed solution X_{SR} of the Riccati equation degrades, until it finally stagnates by an error of the order of 10^{-7}.

8. The (Implicitly Restarted) Symplectic Lanczos Method for Hamiltonian Matrices. In this section, first we review the (implicitly restarted) symplectic Lanczos method [17] to compute a certain subset of the eigenvalues of a Hamiltonian matrix H. In [17] only a single shift implicitly restarted Lanczos algorithm is considered. The double and quadruple shift case has been considered in [132]. As, unfortunately, the discussion of the properties of the algorithm given in [17] is incorrect, a corrected version is given here. Moreover, a Krylov-Schur-like restart is presented. Numerical examples are given.

The usual nonsymmetric Lanczos algorithm generates two sequences of vectors. Due to the Hamiltonian structure of H it is easily seen that one of the two sequences can be eliminated here and thus work and storage can essentially be halved. (This property is valid for a broader class of matrices, see [61].)

The structure-preserving symplectic Lanczos method [17, 57] generates a sequence of matrices that satisfy the Lanczos recursion

$$HS^{2n,2k} = S^{2n,2k}\tilde{H}^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T. \tag{8.1}$$

Here, $\tilde{H}^{2k,2k}$ is a Hamiltonian J-Hessenberg matrix. The space spanned by the columns of $S^{2n,2k}$ is symplectic since ${S^{2n,2k}}^T J^{2n,2n} S^{2n,2k} = J^{2k,2k}$ where $J^{2j,2j}$ is a $2j \times 2j$ matrix of the form (2.1).

In order to simplify the notation in the following we use, as in [38, 64], a permuted version of H and $\widetilde{H}$. Let

$$H_P = PHP^T, \quad \widetilde{H}_P = P\widetilde{H}P^T, \quad S_P = PSP^T, \quad J_P = PJP^T$$

with the permutation matrix $P = P^{2n,2n}$ as in (2.2). To simplify the following discussion, we will assume unless stated otherwise that H is nonderogatory, that is, that each eigenvalue has unit geometric multiplicity.

¿From $S^T JS = J$ we obtain

$$S_P^T J_P S_P = J_P = \mathrm{diag}(\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix})$$

while $S^{-1}HS = \widetilde{H}$ yields $H_P S_P = S_P \widetilde{H}_P$ where

$$\widetilde{H}_P = \left[\begin{array}{cc|cc|cc|cc|cc} \delta_1 & \beta_1 & 0 & \zeta_2 & & & & & & \\ \nu_1 & -\delta_1 & 0 & 0 & & & & & & \\ \hline 0 & \zeta_2 & \delta_2 & \beta_2 & 0 & \zeta_3 & & & & \\ 0 & 0 & \nu_2 & -\delta_2 & 0 & 0 & & & & \\ \hline & & 0 & \zeta_3 & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & 0 & \zeta_n \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & \zeta_n & \delta_n & \beta_n \\ & & & & & & 0 & 0 & \nu_n & -\delta_n \end{array}\right]. \tag{8.2}$$

Hence, the structure–preserving Lanczos method generates a sequence of matrices

$$S_P^{2n,2k} = [v_1, w_1, v_2, w_2, \ldots, v_k, w_k] \in \mathbb{R}^{2n \times 2k}$$

satisfying

$$H_P S_P^{2n,2k} = S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T \tag{8.3}$$

where $\widetilde{H}_P^{2k,2k} = P^k \widetilde{H}^{2k,2k} {P^k}^T$ is a permuted $2k \times 2k$ Hamiltonian J–Hessenberg matrix $\widetilde{H}^{2k,2k}$ of the form (8.2). The space spanned by the columns of $S^{2n,2k} = {P^n}^T S_P^{2n,2k} P^k$ is symplectic since ${S_P^{2n,2k}}^T J_P^{2n,2n} S_P^{2n,2k} = J_P^{2k,2k}$ where $P^j {J^{2j,2j}} {P^j}^T = J_P^{2j,2j}$ and $J^{2j,2j}$ is a $2j \times 2j$ matrix of the form (1.2).

The vector $r_{k+1} := \zeta_{k+1} v_{k+1}$ is the residual vector and is J_P–orthogonal to the columns of $S_P^{2n,2k}$, the permuted Lanczos vectors. The matrix $H_P^{2k,2k}$ is the J_P–orthogonal projection of H_P onto the range of $S_P^{2n,2k}$,

$$H_P^{2k,2k} = J_P^{2k,2k} (S_P^{2n,2k})^T J_P H_P S_P^{2n,2k}.$$

Equation (8.3) defines a length $2k$ Lanczos factorization of H_P. If the residual vector r_{k+1} is the zero vector, then equation (8.3) is called a truncated Lanczos factorization when $k < n$. Note that r_{n+1} must vanish since $(S_P^{2n,2n})^T J_P r_{n+1} = 0$ and the columns of $S_P^{2n,2n}$ form a J_P–orthogonal basis for $\mathbb{R}^{2n}$. In this case the symplectic Lanczos method computes a reduction to permuted J-Hessenberg form.

REMARK 8.1. *The Implicit-S-Theorem 2.12 says that if there are two symplectic Lanczos recurrences*

$$\begin{aligned} H_P S_P^{2n,2k} &= S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T \\ H_P \widehat{S}_P^{2n,2k} &= \widehat{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \breve{\zeta}_{k+1} \widehat{v}_{k+1} e_{2k}^T \end{aligned}$$

then there exists a symplectic diagonal matrix

$$D_k = \begin{bmatrix} C & \\ & C^{-1} \end{bmatrix}, \qquad C \in \mathbb{R}^{k \times k}$$

such that

$$S_P^{2n,2k} = \widehat{S}_P^{2n,2k} D_k.$$

Hence, the symplectic Lanczos factorization is, up to multiplication by a trivial matrix, specified by the starting vector v_1. Hence, as this reduction is strongly dependent on the first column of the transformation matrix that carries out the reduction, we must expect breakdown or near–breakdown in the Lanczos process as they also occur in the reduction process to Hamiltonian J–Hessenberg form. Assume that no such breakdowns occur, and let $S_P = [v_1, w_1, v_2, w_2, \ldots, v_n, w_n]$. For a given v_1, a Lanczos method constructs the matrix S_P columnwise from the equations

$$H_P S_P e_j = S_P \widetilde{H}_P e_j, \qquad j = 1, 2, \ldots \ .$$

That is, for odd numbered columns

$$\begin{aligned} H_P v_{m+1} &= \delta_{m+1} v_{m+1} + \nu_{m+1} w_{m+1} \\ \Longleftrightarrow \nu_{m+1} w_{m+1} &= H_P v_{m+1} - \delta_{m+1} v_{m+1} \\ &=: \widetilde{w}_{m+1} \end{aligned} \tag{8.4}$$

and for even numbered columns

$$\begin{aligned} H_P w_m &= \zeta_m v_{m-1} + \beta_m v_m - \delta_m w_m + \zeta_{m+1} v_{m+1} \\ \Longleftrightarrow \zeta_{m+1} v_{m+1} &= H_P w_m - \zeta_m v_{m-1} - \beta_m v_m + \delta_m w_m \\ &=: \widetilde{v}_{m+1}. \end{aligned} \tag{8.5}$$

Now we have to choose ν_{m+1}, ζ_{m+1} such that $S_P^T J_P S_P = J_P$ is satisfied, that is we have to choose ν_{m+1}, ζ_{m+1} such that $v_{m+1}^T J_P w_{m+1} = 1$. One possibility is to choose

$$\zeta_{m+1} = ||\widetilde{v}_{m+1}||_2, \qquad \nu_{m+1} = v_{m+1}^T J_P H_P v_{m+1}.$$

Premultiplying $\widetilde{v}_{m+1}$ by $w_m^T J_P$ and using $S_P^T J_P S_P = J_P$ yields

$$\beta_m = -w_m^T J_P H_P w_m.$$

¿From this we obtain the algorithm given in Table 1. There is still some freedom in the choice of the parameters that occur in this algorithm. Possibilities to remove these ambiguities have been discussed in [97]. Essentially, the parameters δ_m can be chosen freely. Different choices have been proposed in the literature: $\delta_m = 1$ in [17],

Algorithm : Symplectic Lanczos method

Choose an initial vector $\widetilde{v}_1 \in \mathbb{R}^{2n}, \widetilde{v}_1 \neq 0$.
Set $v_0 = 0 \in \mathbb{R}^{2n}$.
Set $\zeta_1 = ||\widetilde{v}_1||_2$ and $v_1 = \frac{1}{\zeta_1}\widetilde{v}_1$.

for m = 1, 2, ... do

(update of w_m)

set

$\delta_m = v_m^T H_P v_m$

$\widetilde{w}_m = H_P v_m - \delta_m v_m$

$\nu_m = v_m^T J_P H_P v_m$

$w_m = \frac{1}{\nu_m}\widetilde{w}_m$

(computation of β_m)

$\beta_m = -w_m^T J_P H_P w_m$

(update of v_{m+1})

$\widetilde{v}_{m+1} = H_P w_m - \zeta_m v_{m-1} - \beta_m v_m + \delta_m w_m$

$\zeta_{m+1} = ||\widetilde{v}_{m+1}||_2$

$v_{m+1} = \frac{1}{\zeta_{m+1}}\widetilde{v}_{m+1}$

end

TABLE 8.1
Symplectic Lanczos Method

$\delta_m = 0$ in [139], or $\delta_m = v_m^T H_P v_m$ in [57]. The last choice of δ_m implies that the Lanczos vectors v_m and w_m are orthogonal to each other. Likewise a different choice of the parameters ζ_m, ν_m is possible.

REMARK 8.2. *In case $\delta_m = 0$ is chosen for all m, the symplectic Lanczos algorithm applied to H_P is equivalent to the nonsymmetric Lanczos algorithm applied to the skew-Hamiltonian matrix H_P^2, but costs half as much to execute. The nonsymmetric Lanczos algorithm is a structure-preserving algorithm for skew-Hamiltonian matrices. See [139] for details.*

REMARK 8.3. [3] *In general, the accuracy of the computed eigenvalues is related to the condition number of the transformation matrix which transforms the given matrix to a form from which the eigenvalues can be read off. Hence, if the Lanczos process is run to completion such that $H_P S_P = S_P \widetilde{H}_P$, then S_P is this transformation matrix. Therefore, the condition number $\kappa(S_P)$ should be kept as small as possible. As S_P is a permuted symplectic matrix, we have $\kappa(S_P) = \kappa(S)$ and as $S^{-1} = -JS^TJ$*

$$\kappa(S) = ||S||_2^2.$$

That is, when choosing the free parameters, we should aim at minimizing $||S||_2$. As

$$||S||_2 \leq 2n \max_{1 \leq k \leq n} \{||v_k||_2, ||w_k||_2\}$$

[3] This remark is due to an unpublished manuscript by Peter Benner.

a suboptimal choice is to minimize the upper bound. If we require $||v_k||_2 = 1$ *(as in [17, 57, 139], this reduces to*

$$||S||_2 \leq 2n \max_{1\leq k\leq n}\{||w_k||_2\} = 2n \max_{1\leq k\leq n}\{\frac{1}{|\nu_k|}||H_P v_k - \delta_k v_k||_2\}.$$

In order to obtain a symplectic basis, ν_k *is not a free parameter. Hence, the only free parameter left is* δ_k *and we can minimize the right-hand side of the above expression by*

$$\min_{\delta_1,\ldots,\delta_k} \max_{1\leq k\leq n}\{\frac{1}{|\nu_k|}||H_P v_k - \delta_k v_k||_2\}.$$

As the choice of δ_k *doe snot influence the Lanczos vectors* v_j, w_j *for* $1 \leq j \leq k-1$*, this min-max-problem is simply solved by solving in each Lanczos step*

$$\delta_k = \text{argmax}_{\delta\in\mathbb{R}}\{||H_P v_k - \delta v_k||_2\}.$$

The corresponding quadratic form

$$f_k(\delta) = ||H_P v_k - \delta v_k||_2^2 = v_k^T H_P^T H_P v_k - 2\delta v_k^T H_P v_k + \delta v_k^T v_k$$

yields the following first-order necessary condition for a minimum (as $v_k^T v_k = 1$*)*

$$f'(\delta) = -2v_k^T H_P v_k + 2\delta = 0.$$

Hence, the optimum is

$$\delta_k = v_k^T H_P v_k.$$

This is the local orthogonality condition proposed first in [57].

Note that only one matrix–vector product is required for each computed Lanczos vector w_m or v_m. Thus an efficient implementation of this algorithm requires $6n + (4nz + 32n)k$ flops where nz is the number of nonzero elements in H_P and $2k$ is the number of Lanczos vectors computed (that is, the loop is executed k times). The algorithm as given in Table 1 computes an odd number of Lanczos vectors, for a practical implementation one has to omit the computation of the last vector v_{k+1} (or one has to compute an additional vector w_{k+1}).

In the symplectic Lanczos method as given above we have to divide by a parameter that may be zero or close to zero. If such a case occurs for the normalization parameter ζ_{m+1}, the corresponding vector $\widetilde{v}_{m+1}$ is zero or close to the zero vector. In this case, a symplectic invariant subspace of H (or a good approximation to such a subspace) is detected. By redefining $\widetilde{v}_{m+1}$ to be any vector satisfying

$$v_j^T J_P \widetilde{v}_{m+1} = 0$$
$$w_j^T J_P \widetilde{v}_{m+1} = 0$$

for $j = 1, \ldots, m$, the algorithm can be continued. The resulting Hamiltonian J–Hessenberg matrix is no longer unreduced; the eigenproblem decouples into two smaller subproblems. In case $\widetilde{w}_m$ is zero (or close to zero), an invariant subspace of H_P with dimension $2m-1$ is found (or a good approximation to such a subspace). It is easy to see that in this case the parameter ν_m will be zero (or close to zero).

Two eigenvalues and one right and one left eigenvector can be read off directly from the reduced matrix $\widetilde{H}_P^{2m-2,2m-2}$ (8.2). We obtain from Table 8.1 that in this case $H_P v_m = \delta_m v_m$, i.e., δ_m is an eigenvalue of H_P with corresponding eigenvector v_m. (In case $\widetilde{w}_m \approx 0$, we have $H_P v_m \approx \delta_m v_m$). Due to the symmetry of the spectrum of H, we also have that $-\delta_m$ is an eigenvalue of H. Computing an eigenvector y of H_P corresponding to $-\delta_m$, we can try to augment the $(2m-1)$–dimensional invariant subspace to an H_P–invariant subspace of even dimension. If this is possible, the space can be made J_P–orthogonal by J_P–orthogonalizing y against $\{ v_1, w_1, \ldots, v_{m-1}, w_{m-1} \}$ and normalizing such that $y^T J_P v_m = 1$.

Thus if either v_{m+1} or w_{m+1} vanishes, the breakdown is benign. If $v_{m+1} \neq 0$ and $w_{m+1} \neq 0$ but $\nu_{m+1} = 0$, then the breakdown is serious. No reduction of the Hamiltonian matrix to a Hamiltonian J–Hessenberg matrix with v_1 as first column of the transformation matrix exists. On the other hand, an initial vector v_1 exists so that the symplectic Lanczos process does not encounter serious breakdown. However, determining this vector requires knowledge of the minimal polynomial of H. Thus, no algorithm for successfully choosing v_1 at the start of the computation yet exists.

It is well-known that in the symmetric Lanczos procedure, loss of orthogonality between the computed Lanczos vectors signals the convergence of an eigenpair, see, e.g. [107]. In [8] it is shown that this is also the case in the non-symmetric eigenvalue problem. The cancellation error and resultant loss of orthogonality appears under similar conditions as for the symmetric problem, and is controlled by the conditioning of the eigenvalue. A similar statement can be shown for the symplectic Lanczos algorithm discussed here, see [57] and [58]. Moreover, the rounding error analysis of the non-symmetric Lanczos process in finite-precision arithmetic as in [8] can be carried over to the symplectic Lanczos process, see [57].

Without some form of re–J–orthogonalization the symplectic Lanczos method is numerically unstable (see Section 8.6.1 and the discussion there). Thus, the symplectic Lanczos method suffers from the same numerical difficulties as any other Lanczos–like algorithm.

The numerical difficulties of the symplectic Lanczos method described above are inherent to all Lanczos-like methods for nonsymmetric matrices. Different approaches to overcome these difficulties have been proposed. Taylor [133] and Parlett, Taylor, and Liu [112] were the first to propose a look-ahead Lanczos algorithm that skips over breakdowns and near-breakdowns. Freund, Gutknecht, and Nachtigal present in [63] a look-ahead Lanczos code that can handle look-ahead steps of any length. Freund and Mehrmann adapted this method to the symplectic Lanczos method given in [64]. The price paid is that the resulting matrix is no longer of J-Hessenberg form, but has a small bulge in the form to mark each occurrence of a (near) breakdown. Unfortunately, so far there exists no eigenvalue method that can make use of that special reduced form.

A different approach to deal with the numerical difficulties of Lanczos-like algorithms is to implicitly restart the symplectic Lanczos factorization. This was first introduced by Sorensen [126] in the context of nonsymmetric matrices and the Arnoldi process. Usually only a small subset of the eigenvalues is desired. As the eigenvalues of the Hamiltonian J-Hessenberg matrices $H^{2k,2k}$ are estimates for the eigenvalues of H, the length $2k$ symplectic Lanczos factorization (8.3) may suffice if the residual vector r_{k+1} is small. The idea of restarted Lanczos algorithms is to fix the number of steps in the Lanczos process at a prescribed value k which is dependent on the required number of approximate eigenvalues. The purpose of the implicit restart is

to determine initial vectors such that the associated residual vectors are tiny. Given (8.3), an implicit Lanczos restart computes the Lanczos factorization

$$H\breve{S}^{2n,2k} = \breve{S}^{2n,2k}\breve{H}^{2k,2k} + \breve{r}_{k+1}e_{2k}^T$$

which corresponds to the starting vector

$$\breve{s}_1 = p(H)s_1$$

(where $p(H) \in \mathbb{R}^{2n\times 2n}$ is a polynomial) without having to explicitly restart the Lanczos process with the vector $\breve{s}_1$. This process is iterated until the residual vector r_{k+1} is tiny. J–orthogonality of the k Lanczos vectors is secured by re–J–orthogonalizing these vectors when necessary. This idea will be investigated in Section 8.4. As the iteration progresses, some of the Ritz values may converge to eigenvalues of H long before the entire set of wanted eigenvalues have. These converged Ritz values may be part of the wanted or unwanted portion of the spectrum. In either case it is desirable to deflate the converged Ritz values and corresponding Ritz vectors from the unconverged portion of the factorization. If the converged Ritz value is wanted then it is necessary to keep it in the subsequent factorizations; if it is unwanted then it must be removed from the current and the subsequent factorizations. A short comment on locking and purging techniques to accomplish this is given in Section 8.4.1. Most of the complications in the purging and deflating algorithms come from the need to preserve the structure of the decomposition, in particular, to preserve the J-Hessenberg form and the zero structure of the vector e_{2k}^T. In [130], Stewart shows how to relax the definition of an Arnoldi decomposition such that the purging and deflating problems can be solved in a natural and efficient way. Since the method is centered about the Schur decomposition of the Hessenberg matrix, the method is called the Krylov-Schur method. In Section 8.5, a Krylov-Schur-like method for the symplectic Lanczos method is developed as first developed in [132].

But first, we will discuss the truncated symplectic Lanczos factorizations in Section 8.1, that is, first we are concerned with finding conditions for the symplectic Lanczos method terminating prematurely. This is a welcome event since in this case we have found an invariant symplectic subspace $S^{2n,2k}$ and the eigenvalues of $H^{2k,2k}$ are a subset of those of H. We will first discuss the conditions under which the residual vector of the symplectic Lanczos factorization will vanish at some step k. Then we will show how the residual vector and the starting vector are related. Finally a result indicating when a particular starting vector generates an exact truncated factorization is given. Stopping criteria which guarantee the required accuracy of the computed Ritz values and vectors are discussed in Section 8.2.

The symplectic Lanczos algorithm described above will, in general, compute approximations to a few of the largest eigenvalues of a Hamiltonian matrix H. Sometimes only a few of its smallest eigenvalues are needed. Since these are also the largest eigenvalues of H^{-1}, a Krylov subspace method can be applied to H^{-1} to find them. Since H^{-1} inherits the Hamiltonian structure of H, the symplectic Lanczos method is an appropriate method in the interest of efficiency, stability and accuracy. In situations where some prior information is given, one might prefer to use a shift before inverting. Specifically, if we know that the eigenvalues of interest lie near τ, we might prefer to work with $(H - \tau I)^{-1}$. Unfortunately, the shift destroys the Hamiltonian structure. Appropriate shift-and-invert strategies are discussed in Section 8.3.

Numerical properties of the symplectic Lanczos algorithm are discussed in the final section of this chapter. Numerical experiments are given.

8.1. Truncated symplectic Lanczos factorizations. This section is concerned with finding conditions for the symplectic Lanczos method terminating prematurely. This is a welcome event since in this case we have found an invariant symplectic subspace $S^{2n,2k}$ and the eigenvalues of $H^{2k,2k}$ are a subset of those of H. We will first discuss the conditions under which the residual vector of the symplectic Lanczos factorization will vanish at some step k. Then we will show how the residual vector and the starting vector are related. Finally a result indicating when a particular starting vector generates an exact truncated factorization is given.

First the conditions under which the residual vector of the symplectic Lanczos factorization will vanish at some step k will be discussed. From the derivation of the algorithm it is immediately clear that if no breakdown occurs, then

$$\begin{aligned}\text{span}\{&v_1, v_2, v_3, \ldots, v_{k+1}, w_1, w_2, \ldots, w_k\}\\ &= \text{span}\{v_1, H_P^2 v_1, H_P^4 v_1, \ldots, H_P^{2k} v_1, H_P v_1, H_P^3 v_1, \ldots, H_P^{2k-1} v_1\}\\ &= \text{span}\{\mathcal{K}(H_P, v_1, 2k)\}\end{aligned}$$

and

$$\begin{aligned}\text{span}\{&v_1, v_2, v_3 \ldots, v_{k+1}, w_1, w_2, \ldots, w_{k+1}\}\\ &= \text{span}\{v_1, H_P^2 v_1, H_P^4 v_1, \ldots, H_P^{2k} v_1, H_P v_1, H_P^3 v_1, \ldots, H_P^{2k+1} v_1\}\\ &= \text{span}\{\mathcal{K}(M_P, v_1, 2k+1)\},\end{aligned}$$

where $\mathcal{K}(X, v, j) = \{v, Xv, X^2 v \ldots, X^j v\}$. Clearly, dim $\mathcal{K}(H_P, v_1, j) \leq j+1$. Further it is easy to see that

$$\text{dim } \mathcal{K}(H_P, v_1, \ell) = d < \ell \quad \Longrightarrow \quad \text{dim } \mathcal{K}(H_P, v_1, j) = d \quad \forall\ j > \ell. \tag{8.6}$$

If dim $\mathcal{K}(H_P, v_1, 2k+1) = 2k+1$, then

$$H_P v_{k+1} \in \text{span}\{v_1, \ldots, v_{k+1}, w_1, \ldots, w_k\}.$$

Hence, there exist real scalars $a_1, \ldots, a_{k+1}$ and $b_1, \ldots, b_k$ such that

$$H_P v_{k+1} = a_1 v_1 + \ldots + a_k v_{k+1} + b_1 w_1 + \ldots + b_k w_k.$$

Using the definition of ν_{k+1} as given in Table 8.1 and the above expression we obtain because of J–orthogonality,

$$\begin{aligned}\nu_{k+1} &= v_{k+1}^T J_P H_P v_{k+1}\\ &= a_1 v_{k+1}^T J_P v_1 + \ldots + a_{k+1} v_{k+1}^T J_P v_{k+1} + b_1 v_{k+1}^T J_P w_1 + \ldots + b_k v_{k+1}^T J_P w_k\\ &= 0.\end{aligned}$$

As $\widetilde{w}_{k+1} = \nu_{k+1} w_{k+1} = H_P v_{k+1} - \delta_{k+1} v_{k+1}$ (see Table 8.1) it follows that $\widetilde{w}_{k+1} = 0$. This implies that an invariant subspace of H_P with dimension $2k+1$ is found.

If dim $\mathcal{K}(H_P, v_1, 2k) = 2k$, then $H_P w_k \in \text{span}\{v_1, \ldots, v_k, w_1, \ldots, w_k\}$. Hence

$$H_P w_k = a_1 v_1 + \ldots + a_k v_k + b_1 w_1 + \ldots + b_k w_k,$$

for properly chosen a_j, b_j and from the algorithm in Table 8.1

$$\begin{aligned}\widetilde{v}_{k+1} = a_1 v_1 + \ldots + a_{k-2} v_{k-2} + (a_{k-1} - \zeta_k) v_{k-1} + (a_k - \beta_k) v_k\\ + b_1 w_1 + \ldots + b_{k-1} w_{k-1} + (b_k + \delta_k) w_k.\end{aligned}$$

Since $[v_1, w_1, \ldots, v_k, w_k]^T J_P \widetilde{v}_{k+1} = [0, \ldots, 0]$ we obtain for $j < k$ and $\ell < k-2$

$$\begin{aligned}
v_j^T J_P \widetilde{v}_{k+1} &= b_j v_j^T J_P w_j = b_j = 0 \\
v_k^T J_P \widetilde{v}_{k+1} &= (b_k + \delta_k) v_k^T J_P w_k = b_k + \delta_k = 0 \\
w_\ell^T J_P \widetilde{v}_{k+1} &= a_\ell w_\ell^T J_P v_\ell = -a_\ell = 0 \\
w_{k-1}^T J_P \widetilde{v}_{k+1} &= (a_{k-1} - \zeta_k) w_{k-1}^T J_P v_{k-1} = \zeta_k - a_{k-1} = 0 \\
w_k^T J_P \widetilde{v}_{k+1} &= (a_k - \beta_k) w_k^T J_P v_k = \beta_k - a_k = 0.
\end{aligned}$$

Therefore $\widetilde{v}_{k+1} = 0$ and further $\zeta_{k+1} = 0$. This implies that the residual vector of the symplectic Lanczos factorization will vanish at the first step k such that the dimension of $\mathcal{K}(H, v_1, 2k)$ is equal to $2k$ and hence is guaranteed to vanish for some $k \leq n$.

Next we will discuss the relation between the residual term and the starting vector. Here, $\hat{v}_1$ will denote the first Lanczos vector after permuting it back, i.e., $\hat{v}_1 = P^T v_1$. If $\dim\ \mathcal{K}(H, \hat{v}_1, 2n-1) = 2n$ then

$$HK(H, \hat{v}_1, 2n-1) = K(H, \hat{v}_1, n) C_n$$

where $K(H, \hat{v}_1, 2n-1) = [\hat{v}_1, H\hat{v}_1, H^2\hat{v}_1, \ldots, H^{2n-1}\hat{v}_1] \in \mathbb{R}^{2n \times 2n}$, and $C_n \in \mathbb{R}^{2n \times 2n}$ is a companion matrix of the form

$$C_n = \begin{bmatrix} 0 & & & & c_0 \\ 1 & 0 & & & \vdots \\ & 1 & \ddots & & \\ & & & 0 & c_{2n-2} \\ & & & 1 & c_{2n-1} \end{bmatrix}.$$

Thus, for $k < n$

$$HK(H, \hat{v}_1, 2k-1) = K(H, \hat{v}_1, 2k-1) C_k + (H^{2k}\hat{v}_1 - K(H, \hat{v}_1, 2k-1) C_k e_{2k}) e_{2k}^T. \quad (8.7)$$

Define the residual in (8.7) by

$$f_{2k+1} := H^{2k}\hat{v}_1 - K(H, \hat{v}_1, 2k-1) C_k e_{2k}. \quad (8.8)$$

Note that

$$f_{2k+1} = p_{2k}(H)\hat{v}_1 \quad (8.9)$$

where

$$p_{2k}(\lambda) := \lambda^{2k} - \sum_{j=0}^{2k-1} c_j \lambda^j.$$

We will now show that f_{2k+1} is up to scaling the residual of the length $2k$ symplectic Lanczos iteration with starting vector $\hat{v}_1$. Together with (8.9) this reveals the relation between residual and starting vectors. Since $\det(\lambda I - C_k) = \lambda^{2k} - \sum_{j=0}^{2k-1} c_j \lambda^j$, we obtain

$$p_{2k}(\lambda) = \det(\lambda I - C_k).$$

Let $K(H, \hat{v}_1, 2k-1) = S^{2n,2k} R$ where $S^{2n,2k} \in \mathbb{R}^{2n \times 2k}$ with J–orthogonal columns (that is, $(S^{2n,2k})^T J^{2n,2n} S^{2n,2k} = J^{2,2k}$) and $R \in \mathbb{R}^{2k \times 2k}$ is a J–triangular matrix.

Then $S^{2n,2k}e_1 = \hat{v}_1$. The diagonal elements of R are nonzero if and only if the columns of $K(H, \hat{v}_1, 2k-1)$ are linear independent. Choosing

$$c = \begin{bmatrix} c_0 \\ \vdots \\ c_{2k-1} \end{bmatrix} = R^{-1}(-J^{2k,2k}(S^{2n,2k})^T J^{2n,2n})H^{2k}\hat{v}_1$$

assures that $(-J^{2k,2k}(S^{2n,2k})^T J^{2,2n})f_{2k+1} = 0$. Now multiplying (8.7) from the right by R^{-1} yields

$$\begin{aligned} HK(H, \hat{v}_1, 2k-1)R^{-1} - K(H, \hat{v}_1, 2k-1)C_k R^{-1} &= f_{2k+1}e_{2k}^T R^{-1} \\ \iff \qquad HS^{2n,2k} - S^{2n,2k}\widetilde{H} &= f_{2k+1}e_{2k}^T / r_{2k,2k} \end{aligned} \tag{8.10}$$

where $\widetilde{H} = RC_kR^{-1}$ is an unreduced J-Hessenberg matrix (see the proof of the Implicit-S-Theorem 2.12) with the same characteristic polynomial as C_k. Equation (8.10) is a valid symplectic Lanczos recursion with starting vector $\hat{v}_1 = S^{2n,2k}e_1$ and residual vector $f_{2k+1}/r_{2k,2k}$. By (8.9) and due to the essential uniqueness of the symplectic Lanczos recursion any symplectic Lanczos recursion with starting vector $\hat{v}_1$ yields a residual vector that can be expressed as a polynomial in H times the starting vector $\hat{v}_1$.

REMARK 8.4. *¿From (8.8) it follows that if* $\dim \mathcal{K}(H, \hat{v}_1, 2k) \leq 2k$*, then we can choose* $c_0, ..., c_{2k-1}$ *such that* $f_{2k+1} = 0$*. This shows that if the Krylov subspace* $\mathcal{K}(H, \hat{v}_1, 2k-1)$ *forms a* $2k$*–dimensional* H*–invariant subspace, the residual of the symplectic Lanczos recursion will be zero after* k *Lanczos steps such that the columns of* $S^{2n,2k}$ *span a symplectic basis for the subspace* $\mathcal{K}(H, \hat{v}_1, 2k-1)$*.*

The final result of this section will give necessary and sufficient conditions for a particular starting vector to generate an exact truncated factorization in a similar way as stated for the Arnoldi method in [126]. This is desirable since then the columns of $S^{2n,2k}$ form a basis for an invariant symplectic subspace of H and the eigenvalues of $\widetilde{H}^{2k,2k}$ are a subset of those of H. Here, $\hat{v}_k$, $\hat{w}_k$ will denote the Lanczos vectors after permuting them back, i.e., $\hat{v}_k = P^T v_k$, $\hat{w}_k = P^T w_k$.

THEOREM 8.5. *Let* $HS^{2n,2k} - S^{2n,2k}\widetilde{H}^{2k,2k} = \zeta_{k+1}\hat{v}_{k+1}e_{2k}^T$ *be the symplectic Lanczos factorization after* k *steps, with* $\widetilde{H}^{2k,2k}$ *unreduced. Then* $\zeta_{k+1} = 0$ *if and only if* $\hat{v}_1 = Xy$ *where* $HX = XY$ *with rank* $(X) = 2k$ *and* Y *a Jordan matrix of order* $2k$*.*

Proof. If $\zeta_{k+1} = 0$, let $\widetilde{H}^{2k,2k}\widetilde{X} = \widetilde{X}Y$ be the Jordan canonical form of $\widetilde{H}^{2k,2k}$ and put $X = S^{2n,2k}\widetilde{X}$. Then $HX = S^{2n,2k}\widetilde{H}^{2k,2k}\widetilde{X} = S^{2n,2k}\widetilde{X}Y = XY$ and $\hat{v}_1 = S^{2n,2k}e_1 = S^{2n,2k}\widetilde{X}\widetilde{X}^{-1}e_1 = Xy$ with $y = \widetilde{X}^{-1}e_1$.

Suppose now that $HX = XY$, $\operatorname{rank}(X) = 2k$ and $\hat{v}_1 = Xy$. Then $H^m X = XY^m$ for $m \in \mathbb{N}$ and it follows that

$$H^m\hat{v}_1 = H^m Xy = XY^m y \in \operatorname{Range}(X)$$

for $m \in \mathbb{N}$. Hence by (8.6) $\dim\ \mathcal{K}(H, \hat{v}_1, 2k+1) \leq \operatorname{rank}\ (X) = 2k$. Since $\widetilde{H}^{2k,2k}$ is unreduced, $\dim\ \mathcal{K}(H, \hat{v}_1, j) = j+1$ for $j = 1, ..., 2k$. Hence $\dim\ \mathcal{K}(H, \hat{v}_1, 2k+1) = 2k$ and therefore, $\zeta_{k+1} = 0$. ∎

A similar result may be formulated in terms of Schur vectors or symplectic Schur vectors (see, e.g., [98, 92] for the real symplectic Schur decomposition of a symplectic matrix). It is known (see, e.g., Theorem 2.15) that for any symplectic matrix $H \in$

$\mathbb{R}^{2n\times 2n}$ which has no purely imaginary eigenvalues, there exists an orthogonal and symplectic matrix Q such that

$$Q^T H Q = \begin{bmatrix} T & N \\ 0 & -T^T \end{bmatrix}, \qquad T, N \in \mathbb{R}^{n\times n}, \tag{8.11}$$

where T is quasi upper triangular. Q can be chosen such that T has only eigenvalues in the open left half plane. Such a symplectic Schur decomposition exists for a broader class of Hamiltonian matrices, see Theorem 2.15.

THEOREM 8.6. *Let H be a Hamiltonian matrix having a symplectic Schur decomposition as in Theorem 2.15. Let*

$$HS^{2n,2k} - S^{2n,2k}H^{2k,2k} = \zeta_{k+1}\hat{v}_{k+1}e_{2k}^T$$

be the symplectic Lanczos factorization after k steps, with $H^{2k,2k}$ unreduced. Then $\zeta_{k+1} = 0$ if and only if $\widehat{v}_1 = Q^{2k}y$ where

$$HQ^{2n,2k} = Q^{2n,2k}\begin{bmatrix} T & N \\ 0 & -T^T \end{bmatrix} = Q^{2n,2k}R$$

with $(Q^{2n,2k})^T Q^{2n,2k} = I^{2k,2k}$, the columns of $Q^{2n,2k}$ are J–orthogonal, and T quasi upper triangular of order k.

Proof. If $\zeta_{k+1} = 0$, then $HS^{2n,2k} = S^{2n,2k}H^{2k,2k}$. Let $H^{2k,2k}Z = ZR$ be a real symplectic Schur decomposition where $Z \in \mathbb{R}^{2k,2k}$ is orthogonal and symplectic and R is of the form (8.11). Then

$$\widehat{v}_1 = S_P^{2n,2k}e_1 = S_P^{2n,2k}ZZ^Te_1 =: Q^{2n,2k}y$$

where $y = Z^Te_1$ and $Q^{2n,2k} = S_P^{2n,2k}Z \in \mathbb{R}^{2n\times 2k}$. Note that $HQ^{2n,2k} = Q^{2n,2k}R$.

Suppose now that

$$HQ^{2n,2k} = Q^{2n,2k}R \quad \text{with} \quad (Q^{2n,2k})^T Q^{2n,2k} = I^{2k,2k},$$

the columns of $Q^{2n,2k}$ are J–orthogonal and R is of the form (8.11). Let $\widehat{v}_1 = Q^{2n,2k}y$ with $y \in \mathbb{R}^{2n,2k}$ arbitrary. Now, for any $m \in \mathbb{N}$, $H^mQ^{2n,2k} = Q^{2n,2k}R^m$ and thus

$$H^m\widehat{v}_1 = H^mQ^{2n,2k}y = Q^{2n,2k}R^my \in \text{Range}(Q^{2n,2k}).$$

Hence by (8.6) dim $\mathcal{K}(H, \hat{v}_1, 2k+1) \le$ rank $(Q^{2n,2k}) = 2k$. Since $\widetilde{H}^{2k,2k}$ is unreduced, dim $\mathcal{K}(H, \hat{v}_1, j) = j + 1$ for $j = 1, ..., 2k$. Hence dim $\mathcal{K}(H, \hat{v}_1, 2k + 1) = 2k$ and therefore, $\zeta_{k+1} = 0$. $\surd$

These theorems provide the motivation for the implicit restart developed in Section 8.4. Theorem 8.5 suggests that one might find an invariant subspace by iteratively replacing the starting vector with a linear combination of approximate eigenvectors corresponding to eigenvalues of interest. Such approximations are readily available through the Lanczos factorization.

8.2. Stopping Criteria. Now assume that we have performed k steps of the symplectic Lanczos method and thus obtained the identity (after permuting back)

$$HS^{2n,2k} = S^{2n,2k}H^{2k,2k} + \zeta_{k+1}\hat{v}_{k+1}e_{2k}^T.$$

If the norm of the residual vector is small, the $2k$ eigenvalues of $H^{2k,2k}$ are approximations to the eigenvalues of H. Numerical experiments indicate that the norm of the residual rarely becomes small by itself. Nevertheless, some eigenvalues of $H^{2k,2k}$ may be good approximations to eigenvalues of H. Let λ be an eigenvalue of $H^{2k,2k}$ with the corresponding eigenvector y. Then the vector $x = S^{2n,2k}y$ satisfies

$$\begin{aligned} ||Hx - \lambda x|| &= ||(HS^{2n,2k} - S^{2n,2k}H^{2k,2k})y|| \\ &= |\zeta_{k+1}|\,|e_{2k}^T y|\,||\hat{v}_{k+1}||. \end{aligned} \tag{8.12}$$

The vector x is referred to as Ritz vector and λ as Ritz value of H. If the last component of the eigenvector y is sufficiently small, the right-hand side of (8.12) is small and the pair $\{\lambda, x\}$ is a good approximation to an eigenvalue-eigenvector pair of H. Note that by Lemma 2.13 $|e_{2k}^T y| > 0$ if $H^{2k,2k}$ is unreduced. The pair (λ, x) is exact for the nearby problem

$$(H - E)x = \lambda x \qquad \text{where} \quad E = \zeta_{k+1}\hat{v}_{k+1}e_k^T(S^{2n,2k})^T J^{2,2n},$$

as

$$\begin{aligned} (H - E)x &= (H - E)S^{2n,2k}y \\ &= S^{2n,2k}H^{2k,2k}y + \zeta_{k+1}\hat{v}_{k+1}e_{2k}^T y - ES^{2n,2k}y \\ &= \lambda x + \zeta_{k+1}\hat{v}_{k+1}e_{2k}^T y - ES^{2n,2k}y. \end{aligned}$$

A small $||E||$ is not sufficient for the Ritz pair $\{\lambda, x\}$ being a good approximation to an eigenvalue-eigenvector pair of H. The explicit formation of the residual $(HS^{2n,2k} - S^{2n,2k}H^{2k,2k})y$ can be avoided when deciding about the numerical accuracy of an approximate eigenpair, one can use the Ritz estimate $|\zeta_{k+1}|\,|e_{2k}^T y|\,||\hat{v}_{k+1}||$ instead.

It is well-known that for non-normal matrices the norm of the residual of an approximate eigenvector is not by itself sufficient information to bound the error in the approximate eigenvalue. It is sufficient however to give a bound on the distance to the nearest matrix to which the given approximation is exact. In the following, we will give a computable expression for the error. Assume that $H^{2k,2k}$ is diagonalizable

$$Y^{-1}H^{2k,2k}Y = \left[\begin{array}{ccc|ccc} -\lambda_1 & & & & & \\ & \ddots & & & & \\ & & -\lambda_k & & & \\ \hline & & & \lambda_1 & & \\ & & & & \ddots & \\ & & & & & \lambda_k \end{array}\right] = \Lambda;$$

Y can be chosen symplectic. Let $X = S^{2n,2k}Y = [x_1 \ldots, x_{2k}]$ and denote the residual term $\zeta_{k+1}\hat{v}_{k+1}$ by $\hat{r}_{k+1}$. Since $HS^{2n,2k} = S^{2n,2k}H^{2k,2k} + \hat{r}_{k+1}e_{2k}^T$, it follows that

$$HS^{2n,2k}Y = S^{2n,2k}YY^{-1}H^{2k,2k}Y + \hat{r}_{k+1}e_{2k}^T Y$$

or $HX = X\Lambda + \hat{r}_{k+1}e_{2k}^T Y$. Thus

$$Hx_i = -\lambda_i x_i + y_{2k,i}\hat{r}_{k+1} \quad \text{and} \quad Hx_{k+i} = \lambda_i x_{k+i} + y_{2k,k+i}\hat{r}_{k+1}$$

for $i = 1, \ldots, k$. From this, we can conclude relation for the left eigenvectors corresponding to $\pm\lambda$. Premultiplying

$$Hx_i = -\lambda_i x_i + y_{2k,i}\hat{r}_{k+1}$$

by J yields

$$JHx_i = -\lambda_i Jx_i + y_{2k,i}J\hat{r}_{k+1}.$$

As H is Hamiltonian

$$(HJ)^T x_i = -\lambda_i Jx_i + y_{2k,i}J\hat{r}_{k+1},$$

and

$$H^T(Jx_i) = \lambda_i(Jx_i) - y_{2k,i}J\hat{r}_{k+1}.$$

¿From this, we conclude

$$(Jx_i)^T H = \lambda_i(Jx_i)^T - y_{2k,i}\hat{r}_{k+1}^T J.$$

Similarly, we obtain

$$(Jx_{k+i})^T H = -\lambda_i(Jx_{k+i})^T + y_{2k,k+i}\hat{r}_{k+1}^T J.$$

Using Theorem 2' of [73] we obtain that $(-\lambda_i, x_i, (Jx_{k+i})^T)$ is an eigen-triplet of $H - F_{-\lambda_i}$ where

$$\begin{aligned} ||F_{-\lambda_i}||_2 &= \max\ \{\frac{||\hat{r}_{k+1}||_2\ |y_{2k,i}|}{||x_i||_2}, \frac{||\hat{r}_{k+1}^T J||_2\ |y_{2k,k+i}|}{||Jx_{k+i}||_2}\} \\ &= \max\ \{\frac{||\hat{r}_{k+1}||_2\ |y_{2k,i}|}{||x_i||_2}, \frac{||\hat{r}_{k+1}||_2\ |y_{2k,k+i}|}{||x_{k+i}||_2}\}. \end{aligned} \tag{8.13}$$

Furthermore, when $||F_{-\lambda_i}||$ is small enough, then

$$|\theta_i + \lambda_j| \leq \text{cond}(-\lambda_j)||F_{-\lambda_i}|| + \mathcal{O}(||F_{-\lambda_i}||^2),$$

where θ_i is an eigenvalue of H and $\text{cond}(-\lambda_j)$ is the condition number of the Ritz value $-\lambda_j$

$$\text{cond}(-\lambda_j) = \frac{||x_i||_2||Jx_{k+i}||_2}{|x_{k+i}^T Jx_i|}.$$

Similarly, we obtain that $\{\lambda_i, x_{k+i}, (Jx_i)^T\}$ is an eigen-triplet of $H - F_{\lambda_i}$ where

$$\begin{aligned} ||F_{\lambda_i}||_2 &= \max_i\ \left\{\frac{||\widehat{r}_{k+1}||_2\ |y_{2k,k+i}|}{||x_{k+i}||_2}, \frac{||\widehat{r}_{k+1}^T J||_2\ |y_{2k,i}|}{||Jx_i||_2}\right\} \\ &= \max_i\ \left\{\frac{||\widehat{r}_{k+1}||_2\ |y_{2k,k+i}|}{||x_{k+i}||_2}, \frac{||\widehat{r}_{k+1}||_2\ |y_{2k,i}|}{||x_i||_2}\right\}. \end{aligned} \tag{8.14}$$

Consequently, as λ_i and $-\lambda_i$ are treated alike,

$$||F_{-\lambda_i}||_2 = ||F_{\lambda_i}||_2.$$

The symplectic Lanczos algorithm should be continued until $||F_{-\lambda_i}||_2$ is small, and until $\text{cond}(-\lambda_j)||F_{-\lambda_i}||_2$ is below a given threshold for accuracy. Note that as in the Ritz estimate, in the criteria derived here the essential quantities are $|\zeta_{k+1}|$ and the last component of the desired eigenvectors $|y_{2k,i}|$ and $|y_{2k,k+i}|$.

8.3. Shift-and-invert techniques for the symplectic Lanczos method. As noted before, eigenvalues of real Hamiltonian matrices occur in pairs $\{\lambda, -\lambda\}$ or in quadruples $\{\lambda, -\lambda, \overline{\lambda}, -\overline{\lambda}\}$. A structure-preserving algorithm will extract entire pairs and quadruples intact. The symplectic Lanczos algorithm described above will, in general, compute approximations to a few of the largest eigenvalues of a Hamiltonian matrix H. Sometimes only a few of its smallest eigenvalues are needed. Since these are also the largest eigenvalues of H^{-1}, a Krylov subspace method can be applied to H^{-1} to find them. Since H^{-1} inherits the Hamiltonian structure of H, the symplectic Lanczos method is an appropriate method in the interest of efficiency, stability and accuracy. In situations where some prior information is given, one might prefer to use a shift before inverting. Specifically, if we know that the eigenvalues of interest lie near τ, we might prefer to work with $(H - \tau I)^{-1}$. Unfortunately, the shift destroys the Hamiltonian structure. In light of the symmetry of the spectrum, one might think of working with $(H - \tau I)^{-1}(H + \tau I)^{-1}$, in case τ is real or purely imaginary. All eigenvalues near to $\pm\tau$ are mapped simultaneously to values of large modulus. But this matrix is not Hamiltonian as well, it is skew-Hamiltonian. The Cayley transform $(H - \tau I)^{-1}(H + \tau I)$ might come to mind next, but this matrix is symplectic. In case we would like to stay within the Hamiltonian structure, we can work with the Hamiltonian matrix

$$H_1 = H^{-1}(H - \tau I)^{-1}(H + \tau I)^{-1} = (H^3 - \tau^2 H)^{-1},$$

or

$$H_2 = H(H - \tau I)^{-1}(H + \tau I)^{-1} = (H - \tau^2 H^{-1})^{-1},$$

for example. In order to obtain the eigenvalues λ of H from the eigenvalues ϖ of these shifted Hamiltonian matrices, a cubic polynomial equation

$$\lambda^3 - \tau^2\lambda - \frac{1}{\varpi} = 0$$

has to be solved in case H_1 is used, while a quadratic polynomial equation

$$\lambda^2 - \frac{1}{\varpi}\lambda - \tau^2 = 0 \tag{8.15}$$

has to be solved in case H_2 is used. In case a complex shift σ is used, we can work with the Hamiltonian matrix

$$H_3 = H^{-1}(H-\sigma I)^{-1}(H+\sigma I)^{-1}(H-\overline{\sigma} I)^{-1}(H+\overline{\sigma} I)^{-1} = (H^5-(\overline{\sigma}^2+\sigma^2)H^3+|\sigma|^4 H)^{-1}$$

or

$$H_4 = H(H-\sigma I)^{-1}(H+\sigma I)^{-1}(H-\overline{\sigma} I)^{-1}(H+\overline{\sigma} I)^{-1} = (H^3-(\overline{\sigma}^2+\sigma^2)H+|\sigma|^4 H^{-1})^{-1}.$$

Similar as before, in order to obtain the eigenvalues λ of H from the eigenvalues of the shifted matrices, polynomial equations of order five or four have to be solved: in case H_3 is used,

$$\lambda^5 - (\overline{\sigma}^2 + \sigma^2)\lambda^3 + |\sigma|^4\lambda - \frac{1}{\varpi} = 0$$

has to be solved, in case H_4 is used,

$$\lambda^4 - (\overline{\sigma}^2 + \sigma^2)\lambda^2 - \frac{1}{\varpi}\lambda + |\sigma|^4 = 0.$$

Let us consider the case H_2 more closely. The eigenvalues λ of H are mapped to

$$\varpi = \frac{\lambda}{\lambda^2 - \tau^2}.$$

No matter whether $\tau \in \mathbb{R}$ or $\tau \in i\mathbb{R}$, τ^2 is always real. Hence, a real λ is mapped onto a real ϖ, a purely imaginary λ onto a purely imaginary φ and a complex λ onto a complex ϖ. Eigenvectors stay invariant, $Hx = \lambda x$ implies $H^{-1}x = \frac{1}{\lambda}x$ and therefore $H_2 x = \varpi x$ as

$$H_2^{-1}x = (H - \tau^2 H^{-1})x = (\lambda - \frac{\tau^2}{\lambda})x = \frac{1}{\varpi}x.$$

Unfortunately, two distinct eigenvalues λ_1 and λ_2 of H can be mapped to the same eigenvalue ϖ of H_2 by an unlucky choice of τ. Whenever $\tau^2 = -\lambda_1\lambda_2$ is chosen, this is the case (Please note, that τ can be real or purely imaginary, hence τ^2 can be a negative real. Moreover, λ_1 and λ_2 might both be real or purely imaginary, hence the above equation can be fulfilled.)

Applying the symplectic Lanczos process to H_2 yields eigenvalues of the matrix H_2, but we actually want to compute eigenvalues of H. A straightforward approach to compute the eigenvalues λ of H from the eigenvalues ϖ of H_2 is to solve the quadratic equation (8.15). It has the solution

$$\lambda_{1,2} = \frac{1}{2\varpi} \pm \sqrt{\frac{1}{4\varpi^2} + \tau^2}. \tag{8.16}$$

Unfortunately, only one of these solutions corresponds to an eigenvalue of H. In order to decide which one is correct, let us assume that the symplectic Lanczos process is run to achieve a negligible ζ_{k+1},

$$(H_2)_P S_P^{2n,2k} \approx S_P^{2n,2k} \widetilde{H}_P^{2k,2k} \tag{8.17}$$

(and, that H is nonderogatory). The space spanned by the vectors $\{v_1, w_1, \ldots, v_k, w_k\}$ is, up to rounding errors, an invariant subspace under $(H_2)_P$. Normally it is also invariant under H_P as H_2 is a function of H. The space spanned by $\{v_1, w_1, \ldots, v_k, w_k\}$ can fail to be invariant under H only if two distinct eigenvalues of H are mapped to the same eigenvalue of H_2.

Let us assume for the moment that the shift τ is chosen such that this does not happen; that is $\tau^2 \neq -\lambda_1\lambda_2$ for all eigenvalues λ_1, λ_2 of H. If the SR algorithm is used to compute the eigenvalues and eigenvectors of $\widetilde{H}_P^{2k,2k}$

$$\widetilde{H}_P^{2k,2k} \widetilde{S}_P = \widetilde{S}_P \widehat{H}_P,$$

then the eigenvalues λ_j of H can be obtained via (8.16). In order to decide, which of the two possible solutions to choose, we can now check the residual

$$||H_P \widehat{s}_j - \widehat{s}_j \lambda_j||_F$$

where $\widehat{s}_j$ denotes the jth column of $\widehat{S}_P = S_P^{2n,2k}\widetilde{S}_P$ and λ_j has been obtained using the eigenvalue ϖ of $\widehat{H}$ corresponding to the jth column of $\widetilde{S}$. In case the residual is small, the λ_j should be accepted as an eigenvalue of H.

A different approach circumventing the difficulties with the above approach in order to determine the eigenvalues of H from the Lanczos recursion (8.17) is to

calculate the Ritz values of H with respect to the space spanned by the vectors $\{v_1, w_1, \ldots, v_k, w_k\}$ [100]; that is, we calculate the eigenvalues λ_i of

$$X = J_P^{2k,2k}(S_P^{2n,2k})^T J_P H_P S_P^{2n,2k}.$$

As X is Hamiltonian, but not of J-Hessenberg form, it is suggested to compute its eigenvalues by the numerically backward stable structure-preserving HAPACK routine '`haeig`' [23]. Moreover, the residual

$$||H_P S_P^{2n,2k} - S_P^{2n,2k} X||_F$$

can be used to check whether or not the space spanned by the vectors $\{v_1, w_1, \ldots, v_k, w_k\}$ really is invariant under H. Hence, this approach can be used in order to detect if an unlucky shift τ has been chosen.

In order to apply the symplectic Lanczos algorithm to any of the matrices H_j, $j = 1, \ldots, 4$, we need to be able to multiply the matrix H_j by an arbitrary vector at reasonable cost, since this operation is performed repeatedly by the algorithm. Thus,we need to be able to apply operators of the form $(H - \tau I)^{-1}$ inexpensively. The Hamiltonian structure of H should be taken into account here. In case additional information about H is known, this should be made use of as well. If, e.g., H is given in the form

$$H = \begin{bmatrix} A - BC^T & -BB^T \\ CC^T & -(A^T - CB^T) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} + \begin{bmatrix} -B \\ C \end{bmatrix} \begin{bmatrix} C^T & B^T \end{bmatrix}.$$

Such Hamiltonian matrices arise, e.g, in linear quadratic optimal control problems and the solution of continuous-time algebraic Riccati equations [12, 98, 123]. The Sherman-Morrison-Woodbury formula [65] helps in computing the inverse of H:

$$\begin{aligned} H^{-1} &= \begin{bmatrix} A^{-1} & 0 \\ 0 & -A^{-T} \end{bmatrix} \left(I - \begin{bmatrix} -B \\ C \end{bmatrix} (I + F)^{-1} \begin{bmatrix} C^T & B^T \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & -A^{-T} \end{bmatrix} \right) \\ &= \begin{bmatrix} A^{-1} & 0 \\ 0 & -A^{-T} \end{bmatrix} + \begin{bmatrix} \widehat{B} \\ \widehat{C} \end{bmatrix} (I + F)^{-1} \begin{bmatrix} \widehat{C}^T & -\widehat{B}^T \end{bmatrix}, \end{aligned}$$

where

$$\begin{aligned} \widehat{B} &= A^{-1}B, \\ \widehat{C} &= A^{-T}C, \\ F &= \begin{bmatrix} C^T & B^T \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & -A^{-T} \end{bmatrix} \begin{bmatrix} -B \\ C \end{bmatrix} \\ &= \begin{bmatrix} C^T & B^T \end{bmatrix} \begin{bmatrix} -\widehat{B} \\ -\widehat{C} \end{bmatrix} \\ &= -(C^T\widehat{B} + B^T\widehat{C}) = -(C^TA^{-1}B + (C^TA^{-1}B)^T). \end{aligned}$$

Hence, in order to apply H^{-1} to an arbitrary vector, the inverse of A, A^T and $(I - F)$ is needed. Computing the LU decomposition of $A = LU$ beforehand, one can readily apply $A^{-1} = U^{-1}L^{-1}$ and $A^{-T} = L^{-T}U^{-T}$ using the standard backward and forward substitution algorithms. The inverse of $I + F$ is usually easy to compute, as typically in applications like linear quadratic optimal control the matrices B and C only have

a few columns (as even often the multi-input-multi-output systems have only a few inputs and outputs). Hence, $I + F$ often is a small matrix. For a single-input-single-output system, B and C are just vectors, that is, F is a scalar value. The inverse of $I + F$ can easily be obtained.

Using the same approach, we obtain for a shifted matrix $H - \tau I$

$$\begin{aligned}
(H-\tau I)^{-1} &= \left(\begin{bmatrix} (A-\tau I) & 0 \\ 0 & -(A+\tau I)^T \end{bmatrix} + \begin{bmatrix} -B \\ C \end{bmatrix} \begin{bmatrix} C^T & B^T \end{bmatrix} \right)^{-1} \\
&= \left(D + \begin{bmatrix} -B \\ C \end{bmatrix} \begin{bmatrix} C^T & B^T \end{bmatrix} \right)^{-1} \\
&= D^{-1} - D^{-1} \begin{bmatrix} -B \\ C \end{bmatrix} (I+F)^{-1} \begin{bmatrix} C^T & B^T \end{bmatrix} D^{-1} \\
&= \begin{bmatrix} (A-\tau I)^{-1} & 0 \\ 0 & -(A+\tau I)^{-T} \end{bmatrix} + \begin{bmatrix} \widehat{B} \\ \widehat{C} \end{bmatrix} (I+F)^{-1} \begin{bmatrix} \widetilde{C}^T & \widetilde{B}^T \end{bmatrix},
\end{aligned}$$

where

$$\begin{aligned}
\widehat{B} &= (A-\tau I)^{-1} B, \\
\widehat{C} &= (A+\tau I)^{-T} C, \\
\widetilde{B} &= -(A+\tau I)^{-1} B, \\
\widetilde{C} &= (A-\tau I)^{-T} C, \\
F &= \begin{bmatrix} C^T & B^T \end{bmatrix} D^{-1} \begin{bmatrix} -B \\ C \end{bmatrix} \\
&= \begin{bmatrix} C^T & B^T \end{bmatrix} \begin{bmatrix} -\widehat{B} \\ -\widehat{C} \end{bmatrix} \\
&= -(C^T \widehat{B} + B^T \widehat{C}) = -(C^T (A-\tau I)^{-1} B + B^T (A+\tau I)^{-T} C).
\end{aligned}$$

Hence, once the LU decomposition of $(A - \tau I)$ and $(A + \tau I)$ have been determined, $(H - \tau I)^{-1}$ can be applied to any vector in an efficient way.

As

$$(H+\tau I)^{-1} = \left(\begin{bmatrix} (A+\tau I) & 0 \\ 0 & -(A-\tau I)^T \end{bmatrix} + \begin{bmatrix} B \\ -C \end{bmatrix} \begin{bmatrix} C^T & B^T \end{bmatrix} \right)^{-1}$$

no additional LU decomposition are required in order to apply $(H+\tau I)^{-1}$ to a vector.

In case τ is a complex shift, we need to apply $(H-\tau I)^{-1}$, $(H+\tau I)^{-1}$, $(H-\overline{\tau} I)^{-1}$ and $(H+\overline{\tau} I)^{-1}$ to a vector. As H is a real matrix, we have $(H-\overline{\tau} I)^{-1} = \overline{(H-\tau I)}^{-1}$ and $(H+\overline{\tau} I)^{-1} = \overline{(H+\tau I)}^{-1}$. Hence, once the LU decomposition of $(A - \tau I)$ and $(A + \tau I)$ have been determined, all four factors can be applied to any vector in an efficient way.

Once the LU decompositions $A - \tau I$ and $A + \tau I$ are available, the operator H_4 can be set up. This requires

- 16 triangular solves and
- 4 scalar products.

The application of the operator H_4 to a vector now requires

- 16 triangular solves,

- 1 matrix-vector product with A and 1 matrix-vector product with A^T,
- 16 scalar products and
- 16 saxpy operations.

Efficient implementations for computing the required LU decompositions are provided, e.g, by the SuperLU [52], the UMFPACK [49] or the PARDISO package [121]. The ideas presented here have already been used by several authors, see, e.g., [57, 101, 132].

A different class of applications in which large Hamiltonian eigenproblems arise for which the Hamiltonian matrix H allows for an efficient implementation of the shift-and-invert approach are quadratic eigenproblems of the form

$$(\lambda^2 M + \lambda G + K)u = 0, \tag{8.18}$$

where $M, G, K \in \mathbb{R}^{n\times n}$, and $M = M^T > 0, G = -G^T$ and $K = K^T > 0$. It can be shown [7, 101, 135] that the eigenproblem (8.18) has a Hamiltonian eigenstructure, that is, the eigenvalues are symmetric with respect to both axes. In other words, if $\lambda \in \mathbb{C}$ is an eigenvalue with $\mathrm{Re}(\lambda) \neq 0$, then so are $-\lambda, \overline{\lambda}, -\overline{\lambda}$, while if $\lambda \in \mathbb{R}$ or $i\mathbb{R}$ is an eigenvalue, then so is $-\lambda$. It is well known that such quadratic eigenvalue problems can be turned into equivalent generalized ones by suitable linearization. If a structure-preserving linearization is used, the resulting eigenproblem will exhibit the Hamiltonian structure (for a general discussion, see, e.g., [95]). In [7, 101, 135] several linearizations for (8.18) have been proposed. With $y = \lambda M x$ the skew-Hamiltonian/Hamiltonian pencil

$$\lambda\mathcal{N}z - \mathcal{H}z = \lambda \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} - \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = 0$$

is obtained. Here $\mathcal{H}$ is a Hamiltonian matrix, that is and $\mathcal{N}$ is a skew-Hamiltonian matrix, that is

$$(J\mathcal{N})^T = -J\mathcal{N}.$$

Since $\mathcal{N}$ is invertible, the pencil $\lambda\mathcal{N} - \mathcal{H}$ is regular. The skew-Hamiltonian matrix $\mathcal{N}$ can be factorized as

$$\mathcal{N} = \mathcal{Z}\mathcal{Z} = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix}.$$

Thus,

$$\lambda\mathcal{N} - \mathcal{H} = \mathcal{Z}(\lambda I - \mathcal{Z}^{-1}\mathcal{H}\mathcal{Z}^{-1})\mathcal{Z} = \mathcal{Z}(\lambda I - H)\mathcal{Z}$$

with the Hamiltonian matrix $H = \mathcal{Z}^{-1}\mathcal{H}\mathcal{Z}^{-1}$. Since

$$\mathcal{Z}^{-1} = \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix},$$

we have

$$H = \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix}. \tag{8.19}$$

There are other linearizations which yield the same standard eigenvalue problem for the Hamiltonian matrix H [101].

The inverse of H is given by

$$H^{-1} = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & M^{-1} \\ K & 0 \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix},$$

while the inverse of $(H - \tau I)$ is given by

$$(H - \tau I)^{-1} = \begin{bmatrix} I & \frac{1}{2}G + \tau M \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & M \\ -Q(\tau)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G + \tau M \\ 0 & I \end{bmatrix},$$

with $Q(\tau) = \tau^2 M + \tau G + K$. Once a LU decomposition of $Q(\tau)$ is known, $(H - \tau I)^{-1}$ can be applied to a vector in an efficient way. It is easy to see that the same LU decomposition is needed for applying $(H + \tau I)^{-1}$, $(H - \overline{\tau} I)^{-1}$ and $(H + \overline{\tau} I)^{-1}$ to a vector. As $Q(\tau)^T = Q(-\tau)$ we have

$$(H + \tau I)^{-1} = \begin{bmatrix} I & \frac{1}{2}G - \tau M \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & M \\ -Q(\tau)^{-T} & 0 \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G - \tau M \\ 0 & I \end{bmatrix},$$

and as H is a real matrix, we have $(H - \overline{\tau} I)^{-1} = \overline{(H - \tau I)}^{-1}$ and $(H + \overline{\tau} I)^{-1} = \overline{(H + \tau I)}^{-1}$ for a complex τ. Hence, once the LU decomposition of $Q(\tau)$ has been determined, all four factors can be applied to any vector in an efficient way. In case one would like to set up $H_2(\tau)$ or $H_4(\tau)$ we have to multiply by H from (8.19) as well. On first glimpse, one might think that for this the inverse of M is needed. In the following we will see that the multiplication by H essentially comes for free. For this, first consider the skew-Hamiltonian operator

$$R_2(\tau) = (H - \tau I)^{-1}(H + \tau I)^{-1}, \quad \tau \in \mathbb{R}, i\mathbb{R}, \tag{8.20}$$

which is a suitable shift-and-invert operator for the skew-Hamiltonian Arnoldi algorithm SHIRA [101]. It is observed in [101] that this operator can be expressed as

$$\begin{aligned} R_2(\tau) = &\begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \tau I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\tau)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \\ &\times \begin{bmatrix} 0 & I \\ -Q(\tau)^{-T} & 0 \end{bmatrix} \begin{bmatrix} I & -\tau I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix} \end{aligned}$$

A detailed analysis of the cost for applying $R_2(\tau)$ to a vector is provided in [101]. When using $H_2(\tau)$, we have to multiply $R_2(\tau)$ from the left (or the right) by H as given in (8.19). Carefully checking the resulting expression, we obtain that

$$\begin{aligned} H_2(\tau) = &\begin{bmatrix} -\frac{1}{2}G & -K \\ I & 0 \end{bmatrix} \begin{bmatrix} I & \tau I \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & I \\ -Q(\tau)^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \\ &\times \begin{bmatrix} 0 & I \\ -Q(\tau)^{-T} & 0 \end{bmatrix} \begin{bmatrix} I & -\tau I \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix}. \end{aligned}$$

Thus, the only difference in the application of $H_2(\tau)$ as compared to $R_2(\tau)$ is that one multiplication by the mass matrix M is replaced by a multiplication with the stiffness matrix K. As in the applications considered here, the sparsity patterns of M and K are usually the same, the cost for applying $H_2(\tau)$ is the same as for $R_2(\tau)$. In contrast, applying $H_1(\tau)$ requires additionally the inversion of K which makes its application less efficient (as observed in [113]). Similar observations hold for $H_4(\tau)$ compared to $H_3(\tau)$.

Once the LU decompositions of $Q(\tau)$ is available, the operators H_2 and H_4 can be set up. The application of the operator H_2 to a vector now requires

- 4 triangular solves,
- 1 matrix-vector product with M, 1 matrix-vector product with K, 3 matrix-vector products with G, and
- 5 saxpy operations.

The application of the operator H_4 to a vector requires

- 8 triangular solves,
- 3 matrix-vector products with M, 1 matrix-vector product with K, 5 matrix-vector products with G, and
- 9 saxpy operations.

More on the linearization of the quadratic eigenvalue problem and the efficient computation of the matrix-vector products can be found in [7, 101].

8.4. A Restarted Symplectic Lanczos Method. Given that a $2n \times 2k$ matrix S_P^{2k} is known such that

$$H_P S_P^{2n,2k} = S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T \tag{8.21}$$

as in (8.3), an implicit Lanczos restart computes the Lanczos factorization

$$H_P \breve{S}_P^{2n,2k} = \breve{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \breve{\zeta}_{k+1} \breve{v}_{k+1} e_{2k}^T \tag{8.22}$$

which corresponds to the starting vector

$$\breve{v}_1 = p(H_P) v_1$$

for a suitable polynomial p without having to explicitly restart the Lanczos process with the vector $\breve{v}_1$. Such an implicit restarting mechanism will now be derived analogous to the technique introduced in [66, 126].

For reasons of simplicity, let us first consider

$$\breve{v}_1 = \rho(H_P - \mu I) v_1$$

although this single shifted restart will not be used in practice. As eigenvalues of H_P occur in pairs or quadruples, this should be reflected in the implicit restart. For any permuted symplectic $2k \times 2k$ matrix S_P, (8.21) can be re–expressed as

$$H_P(S_P^{2n,2k} S_P) = (S_P^{2n,2k} S_P)(S_P^{-1} \widetilde{H}_P^{2k,2k} S_P) + \zeta_{k+1} v_{k+1} e_{2k}^T S_P.$$

Defining $\breve{S}_P^{2n,2k} = S_P^{2n,2k} S_P$, $\breve{H}_P^{2k,2k} = S_P^{-1} \widetilde{H}_P^{2k,2k} S_P$ this yields

$$H_P \breve{S}_P^{2n,2k} = \breve{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T S_P. \tag{8.23}$$

Let s_{ij} be the (i,j)th entry of S_P. If we choose S_P from the permuted SR decomposition $\widetilde{H}_P^{2k,2k} - \mu I = S_P R_P$, then it is easy to see that S_P is an upper Hessenberg matrix. Thus the residual term in (8.23) is

$$\zeta_{k+1} v_{k+1} (s_{2k,2k-1} e_{2k-1}^T + s_{2k,2k} e_{2k}^T).$$

In order to obtain a residual term of the desired form "vector times e_{2k}^T" we have to truncate off a portion of (8.23). Rewriting (8.23) as

$$H_P \breve{S}_P^{2n,2k} = [\breve{S}_P^{2n,2k-2},\ \breve{v}_k,\ \breve{w}_k,\ v_{k+1}] \left[\begin{array}{c|cc} \breve{H}_P^{2k-2,2k-2} & 0 & \breve{\zeta}_k e_{2k-3} \\ \hline \breve{\zeta}_k e_{2k-2}^T & \breve{\delta}_k & \breve{\beta}_k \\ 0 & \breve{\nu}_k & -\breve{\delta}_k \\ \hline 0 & \zeta_{k+1} s_{2k,2k-1} & \zeta_{k+1} s_{2k,2k} \end{array}\right]$$

we obtain as a new Lanczos identity

$$H_P \breve{S}_P^{2n,2k-2} = \breve{S}_P^{2n,2k-2} \breve{H}_P^{2k-2,2k-2} + \breve{\zeta}_k \breve{v}_k e_{2k-2}^T. \tag{8.24}$$

Here, $\breve{\zeta}_k$, $\breve{\delta}_k$, $\breve{\beta}_k$, $\breve{\nu}_k$ denote parameters of $\breve{H}_P^{2k,2k}$, ζ_{k+1} a parameter of $\widetilde{H}_P^{2k,2k}$. In addition, $\breve{v}_k, \breve{w}_k$ are the last two column vectors from $\breve{S}_P^{2n,2k}$, while v_{k+1} is the next to last column vector of $S_P^{2n,2k}$.

As the space spanned by the columns of $S^{2n,2k} = P^{n^T} S_P^{2n,2k} P^k$ is symplectic, and S_P is a permuted symplectic matrix, the space spanned by the columns of $\breve{S}^{2n,2k-2} = P^{n^T} \breve{S}_P^{2n,2k-2} P^{k-1}$ is symplectic. Thus (8.24) is a valid Lanczos factorization for the new starting vector $\breve{v}_1 = \rho(H_P - \mu I)v_1$. Only one additional step of the symplectic Lanczos algorithm is required to obtain (8.22) from (8.21).

Note that in the symplectic Lanczos process the vectors v_j of $S_P^{2n,2k}$ satisfy the condition $||v_j||_2 = 1$ and the parameters δ_j are chosen such that v_j and w_j are orthogonal. Due to the multiplication by S_P, in general, this is no longer true for the parameters $\breve{\delta}_j$ from $\breve{H}_P^{2k,2k}$ and for the odd numbered column vectors of $\breve{S}_P^{2n,2k}$ and thus for the new Lanczos factorization (8.24).

In case, we choose S_P from the permuted SR decomposition $(\widetilde{H}_P^{2k,2k} - \mu I)(\widetilde{H}_P^{2k,2k} + \mu I) = S_P R_P$ (for $\mu \in \mathbb{R}$ or $\mu = iw, w \in \mathbb{R}$), then S_P is no longer upper Hessenberg, but has an additional lower subdiagonal. Hence the residual term in (8.23) is of the form

$$\zeta_{k+1} v_{k+1} (s_{2k,2k-2} e_{2k-2}^T + s_{2k,2k-1} e_{2k-1}^T + s_{2k,2k} e_{2k}^T).$$

In order to obtain a residual term of the desired form "vector times e_{2k}^T", as before, we have to truncate off a portion of (8.23). Rewriting (8.23) as

$$H_P \breve{S}_P^{2n,2k} = [\breve{S}_P^{2n,2k-4},\ \breve{v}_{k-1},\ \breve{w}_{k-1},\ \breve{v}_k,\ \breve{w}_k,\ v_{k+1}] X$$

where

$$X = \left[\begin{array}{c|c|cc|cc} \breve{H}_P^{2k-4,2k-4} & 0 & \breve{\zeta}_{k-1} e_{2k-5} & & 0 & 0 \\ \hline \breve{\zeta}_{k-1} e_{2k-4}^T & \breve{\delta}_{k-1} & \breve{\beta}_{k-1} & & 0 & \breve{\zeta}_k \\ 0 & \breve{\nu}_{k-1} & -\breve{\delta}_{k-1} & & 0 & 0 \\ \hline 0 & 0 & \breve{\zeta}_k & & \breve{\delta}_k & \breve{\beta}_k \\ 0 & 0 & 0 & & \breve{\nu}_k & -\breve{\delta}_k \\ \hline 0 & 0 & \zeta_{k+1} s_{2k,2k-2} & & \zeta_{k+1} s_{2k,2k-1} & \zeta_{k+1} s_{2k,2k} \end{array}\right]$$

we obtain as a new Lanczos identity

$$H_P \breve{S}_P^{2n,2k-4} = \breve{S}_P^{2n,2k-4} \breve{H}_P^{2k-4,2k-4} + \breve{\zeta}_{k-1} \breve{v}_{k-1} e_{2k-4}^T. \tag{8.25}$$

Similarly, in case, we choose S_P from the permuted SR decomposition $(\widetilde{H}_P^{2k,2k} - \mu I)(\widetilde{H}_P^{2k,2k} + \mu I)(\widetilde{H}_P^{2k,2k} - \overline{\mu} I)(\widetilde{H}_P^{2k,2k} + \overline{\mu} I) = S_P R_P$ (for $\mu \in \mathbb{C}$, $\mathrm{Re}(\mu) \neq 0$), then S_P is no longer upper Hessenberg, but has three additional lower subdiagonals. Hence the residual term in (8.23) is of the form

$$\zeta_{k+1} v_{k+1} (s_{2k,2k-4} e_{2k-4}^T + s_{2k,2k-3} e_{2k-3}^T + s_{2k,2k-2} e_{2k-2}^T + s_{2k,2k-1} e_{2k-1}^T + s_{2k,2k} e_{2k}^T).$$

In order to obtain a residual term of the desired form "vector times e_{2k}^T", as before, we have to truncate off a portion of (8.23). Rewriting (8.23) as

$$H_P \breve{S}_P^{2n,2k} = [\breve{S}_P^{2n,2k-6},\ \breve{v}_{k-2},\ \breve{w}_{k-2},\ \breve{v}_{k-1},\ \breve{w}_{k-1},\ \breve{v}_k,\ \breve{w}_k,\ v_{k+1}] X$$

Algorithm : k–step restarted symplectic Lanczos method

perform k steps of the symplectic Lanczos algorithm to compute $S_P^{2n,2k}$ and $\widetilde{H}_P^{2k,2k}$
while $||\zeta_{k+1}v_{k+1}|| > tol$
perform q additional steps of the symplectic Lanczos method to compute $S_P^{2n,2(k+q)}$ and $\widetilde{H}_P^{2(k+q),2(k+q)}$
select q shifts (keep pairs and quadruples)
obtain $S_P^{2n,2k}$ and $\widetilde{H}_P^{2k,2k}$ from $S_P^{2n,2(k+q)}$ and $\widetilde{H}_P^{2(k+q),2(k+q)}$ by implicit restarts
end

TABLE 8.2
k–step restarted symplectic Lanczos method

where

$$X = \left[\begin{array}{c|cc|cc|cc} \breve{H}_P^{2k-6,2k-6} & 0 & * & 0 & 0 & 0 & 0 \\ \hline \breve{\zeta}_{k-2}e_{2k-6}^T & * & * & 0 & * & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & * & * & * & 0 & * \\ 0 & 0 & 0 & * & * & 0 & 0 \\ \hline 0 & 0 & * & * & * & * & * \end{array}\right]$$

we obtain as a new Lanczos identity

$$H_P\breve{S}_P^{2n,2k-6} = \breve{S}_P^{2n,2k-6}\breve{H}_P^{2k-6,2k-6} + \breve{\zeta}_{k-2}\breve{v}_{k-2}e_{2k-6}^T. \tag{8.26}$$

The extension of this technique to the multiple shift case is straightforward.

The implicitly restarted symplectic Lanczos method will be used to compute a few eigenvalues and associated eigenvectors. For this, we fix the number of steps in the Lanczos process at a prescribed value k of modest size. An attempt will be made to iteratively update the starting vector v_1 by implicit restarts in order to force the residual vector $\zeta_{k+1}v_{k+1}e_{2k}^T$ to zero. That is, we propose the k–step restarted symplectic Lanczos method as given in Table 2 (analogous to [126]).

A detailed discussion of this approach along the lines of the discussion in [126] can be given. The approach has several advantages over the standard symplectic Lanczos method. J–orthogonality can be maintained at reasonable computational costs. There is fixed storage requirement and we can use deflation techniques similar to those associated with the SR iteration.

REMARK 8.7. *In case in the symplectic Lanczos algorithm, δ_m is chosen to be equal to 0 for all m, the implicit restart can be rewritten using the HR instead of the SR algorithm, see [139] for details.*

Numerous choices are possible for the selection of the p shifts. One possibility is to choose p "exact" shifts with respect to $\widetilde{H}_P^{2(k+p),2(k+p)}$. That is, first the eigenvalues of $\widetilde{H}_P^{2(k+p),2(k+p)}$ are computed (by the SR algorithm), then p unwanted eigenvalues are selected. One choice for this selection might be: sort the eigenvalues by decreasing values of the real parts. There will be $k+p$ eigenvalues with nonnegative real parts

$$\begin{aligned} &\text{Re }(\lambda_1) \geq \ldots \geq \text{Re }(\lambda_k) \geq \text{Re }(\lambda_{k+1}) \geq \ldots \geq \text{Re }(\lambda_{k+p}) \geq 0 \\ &\quad \geq -\text{Re }(\lambda_{k+p}) \geq \ldots \geq -\text{Re }(\lambda_{k+1}) \geq -\text{Re }(\lambda_k) \geq \ldots \geq -\text{Re }(\lambda_1). \end{aligned}$$

Select the $2p$ eigenvalues with real part closest to 0 as shifts. If λ_{k+1} is complex with $|\lambda_k| = |\lambda_{k+1}|, \mathrm{Re}\,(\lambda_k) \neq 0$, then we either have to choose $2p+2$ shifts or just $2p-2$ shifts, as λ_{k+1} belongs to a quadruple pair of eigenvalues of $\widetilde{H}_P^{2(k+p),2(k+p)}$ and in order to preserve the symplectic structure either λ_k and λ_{k+1} have to be chosen or none.

A different possibility of choosing the shifts is to keep those eigenvalues that are good approximations to eigenvalues of H. That is, eigenvalues for which (8.12) or (8.13 and (8.14) is small. Again we have to make sure that our set of shifts is complete in the sense described above.

Choosing eigenvalues of $\widetilde{H}_P^{2(k+p),2(k+p)}$ as shifts has an important consequence for the next iterate. Assume for simplicity that $\widetilde{H}_P^{2(k+p),2(k+p)}$ is diagonalizable. Let $\lambda(\widetilde{H}_P^{2(k+p),2(k+p)}) = \{\theta_1, \ldots, \theta_{2k}\} \cup \{\mu_1, \ldots, \mu_{2p}\}$ be a disjoint partition of the spectrum of $\widetilde{H}_P^{2(k+p),2(k+p)}$, where the set of the μ_j contain complete pairs and quadruples of eigenvalues. Selecting the exact shifts $\mu_1, \ldots, \mu_{2p}$ in the implicit restart, yields the shift polynomial

$$p(\lambda) = (\lambda - \mu_1) \cdots (\lambda - \mu_{2p})$$

and

$$p(\widetilde{H}^{2(k+p),2(k+p)}) = S_P R_P.$$

That is, we obtain

$$S_P^{-1} \widetilde{H}^{2(k+p),2(k+p)} S_P = \breve{H}_P^{2(k+p),2(k+p)} = \begin{bmatrix} \breve{H}_P^{2k,2k} & X \\ 0 & Y \end{bmatrix}$$

where $\lambda(\breve{H}_P^{2k,2k}) = \{\theta_1, \ldots, \theta_{2k}\}$ and $\lambda(Y) = \{\mu_1, \ldots, \mu_{2p}\}$. This follows from (4.2). Moreover, the new starting vector has been implicitly replaced by the sum of $2k$ approximate eigenvectors:

$$\breve{v}_1 = S_P^{2n,2(k+p)} S_P e_1 = \frac{1}{\rho} S_P^{2n,2(k+p)} p(\widetilde{H}_P^{2(k+p),2(k+p)}) e_1 = \frac{1}{\rho} S_P^{2n,2(k+p)} \sum_{j=1}^{2k} \zeta_j y_j$$

where $\rho = e_1^T R_P e_1$, $\widetilde{H}_P^{2(k+p),2(k+p)} y_j = \theta_j y_j$, and ζ_j is properly chosen. The last equation follows since $p(\widetilde{H}_P^{2(k+p),2(k+p)}) e_1$ has no component along an eigenvector of $\widetilde{H}_P^{2(k+p),2(k+p)}$ associated with $\mu_j, 1 \leq j \leq 2p$. Hence $\breve{v}_1$ is a linear combination of the $2k$ Ritz vectors associated with the Ritz values that are kept:

$$\breve{v}_1 = \rho \sum_{j=1}^{2k} \zeta_j x_j \qquad \text{where } S_P^{2n,2(k+p)} y_j = x_j.$$

It should be mentioned that the k– step restarted symplectic Lanczos method as in Table 8.2 with exact shifts builds a J–orthogonal basis for a number of generalized Krylov subspaces simultaneously. The subspace of length $2(k+p)$ generated during a restart using exact shifts contains all the Krylov subspaces of dimension $2k$ generated from each of the desired Ritz vectors. We have already seen that after the implicit restart the new starting vector of the Lanczos recursion is a combination of Ritz vectors. Assuming as above that $2p$ exact shifts are used, an induction argument using

the same idea as above shows that the first $2k$ columns of $S_P^{2n,2(k+p)}$ are combinations of the desired $2k$ Ritz vectors. (The only difference to the proof above is in showing that for $2 \leq j \leq 2k$, $S_P e_j$ can be written as $p(\widetilde{H}_P^{2(k+p),2(k+p)})w$ for some vector w. Then $S_P^{2n,2(k+p)} S_P e_j$ is, like $\breve{v}_1$, a combination of the desired Ritz vectors.) Hence, during the next Lanczos run, the subspace of degree $2k$ is $\operatorname{span}\{x_1, \ldots, x_{2k}\}$ where $S_P^{2n,2(k+p)} y_j = x_j$ as above. Let the subspace generated during that run be given by

$$\mathcal{N} = \operatorname{span}\{x_1, \ldots, x_{2k}, v_{k+1}, w_{k+1}, \ldots, v_{k+p}, w_{k+p}\}$$

or equivalently,

$$\mathcal{N} = \operatorname{span}\{x_1, \ldots, x_{2k}, v_{k+1}, H_P v_{k+1}, H_P^2 v_{k+1}, \ldots, H_P^{2p+1} v_{k+1}\}.$$

We will now show that this subspace is equivalent to the subspaces

$$\mathcal{M}_j = \operatorname{span}\{x_1, \ldots, x_{2k}, H_P x_j, \ldots, H_P^{2p+2} x_j\}$$

for all j. The Lanczos run under consideration starts from the equation

$$H_P S_P^{2n,2k} = S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T.$$

For a Ritz vector $x = S_P^{2n,2k} y$ and the corresponding Ritz value λ (that is, $\widetilde{H}_P^{2k,2k} y = \lambda y$) we have

$$H_P x - \lambda x = \zeta_{k+1} v_{k+1} e_{2k}^T y.$$

Hence, with $\alpha = e_{2k}^T y \in \mathbb{R}$ we can rewrite

$$H_P x = \lambda x + \alpha \zeta_{k+1} v_{k+1}. \tag{8.27}$$

Therefore, for all Ritz vectors $x_j, j = 1, \ldots, 2k$, $H_P x_j \in \mathcal{N}$. Then

$$H_P^2 x = \lambda H_P x + \alpha \zeta_{k+1} H_P v_{k+1}.$$

Hence, $H_P^2 x$ is a combination of $H_P x$, and $H_P v_{k+1}$, and therefore for all Ritz vectors $x_j, j = 1, \ldots, 2k$, $H_P^2 x_j \in \mathcal{N}$. Similar for other i, $H_P^i x$ is contained in the subspace $\mathcal{N}$. For example, $H_P^\ell x$ is a linear combination of $H_P^{\ell-1} x$ and $H_P^{\ell-1} v_{k+1}$, and therefore for all Ritz vectors $x_j, j = 1, \ldots, 2k$, $H_P^\ell x_j \in \mathcal{N}$. As $\dim(\mathcal{N}) = \dim(\mathcal{M}_j)$, $\mathcal{N}$ and $\mathcal{M}_j$ span the same space. $\surd$

A similar observation for Sorensen's restarted Arnoldi method with exact shifts was made by Morgan in [104]. For a discussion of this observation see [104] or [89]. Morgan infers *'the method works on approximations to all of the desired eigenpairs at the same time, without favoring one over the other'* [104, p. 1220, l. 7–8 from the bottom]. This remark can also be applied to the method presented here.

Moreover, the implicitly restarted symplectic Lanczos method can be interpreted as a non-stationary subspace iteration. An analogous statement for the implicitly restarted Arnoldi method is given in [87]. Assume that we have computed

$$H_P S_P^{2n,2m} = S_P^{2n,2m} \widetilde{H}_P^{2m,2m} + r_{m+1} e_{2m}^T, \tag{8.28}$$

a length $m = k + p$ symplectic Lanczos reduction. As p shifts for the implicit restart we have chosen $\{\mu_1, \ldots, \mu_p\}$ such that we apply the polynomial

$$p_{2p}(\widetilde{H}) = (\widetilde{H} - \mu_p I)(\widetilde{H} + \mu_p I) \cdot \ldots \cdot (\widetilde{H} - \mu_1 I)(\widetilde{H} + \mu_1 I)$$

during the implicit restart. It is fairly easy to see that

$$p_{2p}(H_P)S_P^{2n,2k} = S_P^{2n,2m} p_{2p}(\widetilde{H}_P^{2m,2m})[e_1, e_2, \ldots, e_{2k}]. \tag{8.29}$$

Applying $p_{2p}(H_P)$ to the first $2k$ columns of $S_P^{2n,2m}$ is equivalent to the basis representation given by the first $2k$ columns of $S_P^{2n,2m} p_{2p}(\widetilde{H}_P^{2m,2m})$. Applying an implicit restart to (8.28) using the spectral function p_{2p}, we essentially apply the SR algorithm with shifts $\pm\mu_1, \ldots, \pm\mu_p$ to $\widetilde{H}_P^{2m,2m}$

$$\widetilde{H}_P^{2m,2m} S_P = S_P \breve{H}_P^{2m,2m}.$$

$S_P \in \mathbb{R}^{2m\times 2m}$ is a symplectic, upper triangular matrix with $m-k$ additional subdiagonals. Write S_P as $S_P = [S_P^{[1]}\ S_P^{[2]}\ S_P^{[3]}]$ with $S_P^{[1]} \in \mathbb{R}^{2m\times 2k}, S_P^{[2]} \in \mathbb{R}^{2m\times 2}, S_P^{[3]} \in \mathbb{R}^{2m\times(2m-2k-2)}$. Then

$$\widetilde{H}_P^{2m,2m} S_P^{[1]} = [S_P^{[1]}\ S_P^{[2]}\ S_P^{[3]}] \begin{bmatrix} \breve{H}_P^{2k,2k} \\ \hline \breve{\zeta}_{k+1} e_{2k}^T \\ \hline 0 \\ \hline 0 \end{bmatrix}.$$

Postmultiplying (8.28) with $S_P^{[1]}$ and using $e_{2m}^T S_P^{[1]} = 0$ which is due to the special form of S_P (upper triangular with $m-k$ additional subdiagonals) we obtain

$$\begin{aligned} H_P S_P^{2n,2m} S_P^{[1]} &= S_P^{2n,2m} \widetilde{H}_P^{2m,2m} S_P^{[1]} + r_{m+1} e_{2m}^T S_P^{[1]} \\ &= \breve{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \breve{\zeta}_{k+1} S_P^{2n,2m} S_P^{[2]} e_1 e_{2k}^T \\ &= \breve{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \breve{r}_{k+1} e_{2k}^T. \end{aligned}$$

where $\breve{S}_P^{2n,2k} = S_P^{2n,2m} S_P^{[1]}$. This is just the implicitly restarted symplectic Lanczos recursion obtained by applying one implicit restart with the polynomial p_{2p}. Applying the SR algorithm with shifts $\pm\mu_1, \ldots, \pm\mu_p$ to $\widetilde{H}_P^{2m,2m}$ is equivalent to computing the permuted SR decomposition

$$p_{2p}(\widetilde{H}_P^{2m,2m}) = S_P R_P.$$

Substituting this into (8.29) we obtain

$$p_{2p}(H_P) S_P^{2n,2k} = S_P^{2n,2m} S_P R_P [e_1, e_2, \ldots, e_{2k}] = \breve{S}_P^{2n,2k} \breve{R}_P,$$

where $\breve{R}_P$ is a $2k \times 2k$ upper triangular matrix. This equation describes a nonstationary subspace iteration. As one step of the implicitly restarted symplectic Lanczos process computes the new subspace spanned by the columns of $\breve{S}_P^{2n,2k}$ from $S_P^{2n,2k}$, the implicitly restarted symplectic Lanczos algorithm can be interpreted as a nonstationary subspace iteration.

In the above discussion we have assumed that the permuted SR decomposition $p(\widetilde{H}_P^{2(k+p),2(k+p)}) = S_P R_P$ exists. Unfortunately, this is not always true. During the bulge-chase in the implicit SR step, it may happen that an ν_j is zero (or almost zero). In that case no reduction to Hamiltonian J-Hessenberg form with the corresponding first column $\breve{v}_1$ does exist. In Section 8.8 we will prove that a serious breakdown in the symplectic Lanczos algorithm is equivalent to such a breakdown of the SR

decomposition. Moreover, it may happen that an element ζ_j is zero (or almost zero) such that

$$\breve{H}_P^{2(k+p),2(k+p)} = \begin{bmatrix} \breve{H}_P^{2j,2j} & \\ & \hat{H}_P \end{bmatrix}.$$

The matrix $\breve{H}_P^{2(k+p),2(k+p)}$ is split, an invariant subspace of dimension $2j$ is found. If $j \geq k$ and all shifts have been applied, then the iteration is halted. Otherwise we can continue as in the procedure described by Sorensen in [126, Remark 3].

8.4.1. Locking and Purging. As the iteration progresses, some of the Ritz values may converge to eigenvalues of H long before the entire set of wanted eigenvalues have. These converged Ritz values may be part of the wanted or unwanted portion of the spectrum. In either case it is desirable to deflate the converged Ritz values and corresponding Ritz vectors from the unconverged portion of the factorization. If the converged Ritz value is wanted then it is necessary to keep it in the subsequent factorizations; if it is unwanted then it must be removed from the current and the subsequent factorizations.

Lehoucq and Sorensen develop in [89, 127] locking and purging techniques to accomplish this in the context of nonsymmetric matrices and the restarted Arnoldi method. A locking operation decouples converged approximate eigenvalues and associated invariant subspaces from the active part of the iteration. A purging operation removes unwanted, but converged eigenpairs. Locking has the effect of isolating an approximate eigenspace once it has converged to a certain level of accuracy and then forcing subsequent Arnoldi vectors to be orthogonal to the converged subspace. With this capability, additional instances of a multiple eigenvalue can be computed to the same specified accuracy without the expense of converging them to unnecessarily high accuracy. Purging allows the deletion of converged but unwanted Ritz values and vectors from the Krylov space when they are not purged naturally by the restarting scheme. With the aid of these deflation schemes, convergence of the implicitly restarted Arnoldi method can be greatly improved. Computational effort is also reduced. A slightly improved variant of those deflation schemes is presented in [128]. In the Arnoldi setting, the projected matrix T is of upper Hessenberg form. Small subdiagonal elements of T may occur during the implicit restarting. However, it is usually the case that there are converged Ritz values appearing in the spectrum of T long before small subdiagonal elements appear. The convergence is usually detected through observation of a small last component in an eigenvector y of T. It turns out that in the case of a small last component of y, there is an orthogonal similarity transformation of T that will give an equivalent Arnoldi factorization with a slightly perturbed T that does indeed have a zero subdiagonal. This is the basis of the deflation scheme used for locking and purging.

The situation for the symplectic Lanczos process is quite similar to the one in the Arnoldi process. It is usually the case that there are converged Ritz values appearing in the spectrum of $\widetilde{H}^{2k,2k}$ long before small subdiagonal elements in the $(1,2)$ block appear. The convergence is usually detected through observation of a small last component in an eigenvector y of $\widetilde{H}^{2k,2k}$. As in the context of an Arnoldi process, it turns out that in the case of a small last component of y, there is a symplectic similarity transformation of $\widetilde{H}^{2k,2k}$ that will give an equivalent symplectic Lanczos factorization with a slightly perturbed $\widetilde{H}^{2k,2k}$ that does indeed have a zero subdiagonal element in the $(1,2)$ block.

As deflating converged Ritz values from an Arnoldi decomposition is a complicated affair, we refrain from presenting the details for the symplectic Lanczos factorization. Instead we will consider a Krylov-Schur-like restarting method (see Section 8.5) as suggested by Stewart [130, 131]. While the implicitly restarted symplectic Lanczos factorization can restart with an arbitrary filter polynomial, the Krylov-Schur-like method cannot do that. When it comes to exact shifts the Krylov-Schur-like method is to be preferred because it is more reliable.

8.5. Hamiltonian Krylov-Schur-type Restarting. Most of the complications in the purging and deflating algorithms described by Lehoucq and Sorensen [89, 127, 128] come from the need to preserve the structure of the decomposition, in particular (for the Arnoldi situation), to preserve the Hessenberg form and the zero structure of the vector e_k^T. In [130], Stewart shows how to relax the definition of an Arnoldi decomposition such that the purging and deflating problems can be solved in a natural and efficient way. Since the method is centered about the Schur decomposition of the Hessenberg matrix, the method is called the Krylov-Schur method. In [132], a Krylov-Schur-like method for the symplectic Lanczos method is developed. Here we follow closely the derivations in [130, 131, 132].

So far, we have considered symplectic Lanczos factorizations of order $2k$ of the form (8.3)

$$H_P S_P^{2n,2k} = S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T.$$

More generally, we will speak of a Hamiltonian Krylov-Schur-type decomposition of order $2k$ if $2k+1$ linearly independent vectors $u_1, u_2, \ldots, u_{2k+1} \in \mathbb{R}^{2n}$ are given such that

$$H_P U_P^{2n,2k} = U_P^{2n,2k} B_P^{2k,2k} + u_{2k+1} b_{2k+1}^T, \tag{8.30}$$

where $U_P^{2n,2k} = [u_1, u_2, \ldots, u_{2k}]$. Equivalently, we can write

$$H_P U_P^{2n,2k} = U_P^{2n,2k+1} \widehat{B}_P^{2k,2k},$$

where $U_P^{2n,2k+1} = [U_P^{2n,2k} \;\; u_{2k+1}]$ and

$$\widehat{B}_P^{2k,2k} = \begin{bmatrix} B_P^{2k,2k} \\ b_{2k+1}^T \end{bmatrix}.$$

This definition removes practically all the restrictions imposed on a symplectic Lanczos decomposition. The vectors of the decomposition are not required to be J_P-orthogonal and the vector b_{2k+1} and the matrix $B_P^{2k,2k}$ are allowed to be arbitrary.

If the columns of $U_P^{2n,2k+1}$ are J_P-orthogonal, we say that the Hamiltonian Krylov-Schur-type decomposition is J_P-orthogonal. Please note, that no particular form of B_P is assumed here. It is uniquely determined by the basis $U_P^{2n,2k+1}$. For if $[V_P^{2n,2k} \;\; v_P]^T$ is any left inverse for $U_P^{2n,2k+1}$, then it follows from (8.30) that

$$B_P^{2k,2k} = (V_P^{2n,2k})^T H_P U_P^{2n,2k}$$

and

$$b_{2k+1}^T = v_P^T H_P U_P^{2n,2k}.$$

In particular, $B_P^{2k,2k}$ is a Rayleigh quotient of H_P.

We say that the Hamiltonian Krylov-Schur-type decomposition spans the space spanned by the columns of $U_P^{2n,2k+1}$. Two Hamiltonian Krylo-Schur-type decompositions spanning the same space are said to be equivalent.

For any nonsingular matrix $Q \in \mathbb{R}^{2k,2k}$ we obtain from (8.30) an equivalent Hamiltonian Krylov-Schur-type decomposition

$$H_P(U_P^{2n,2k}Q) = (U_P^{2n,2k}Q)(Q^{-1}B_P^{2k,2k}Q) + u_{2k+1}(b_{2k+1}^TQ).$$

The two Hamiltonian Krylov-Schur-type decompositions are said to be similar to each other.

If, in (8.30) the vector u_{2k+1} can be written as $u_{2k+1} = \gamma\widehat{u}_{2k+1} + U_P^{2n,2k}a, \gamma \neq 0$, then we have that the Hamiltonian Krylov-Schur-type decomposition

$$H_PU_P^{2n,2k} = U_P^{2n,2k}(B_P + ab_{2k+1}^T) + \gamma\widehat{u}_{2k+1}b_{2k+1}^T$$

is equivalent to the original one, as the space spanned by the columns of $[U_P^{2n,2k} \;\; u_{2k+1}]$ is the same as the space spanned by the columns of $[U_P^{2n,2k} \;\; \widehat{u}_{2k+1}]$.

THEOREM 8.8. *Every Hamiltonian Krylov-Schur-type decomposition is equivalent to a (possibly reduced) symplectic Lanczos factorization.*

Proof. We begin with the Hamiltonian Krylov-Schur-type decomposition

$$H_PU_P = U_PB_P + ub^T,$$

where for convenience we have dropped all sub- and superscripts. Let $U_P = S_PR_P$ be the permuted SR decomposition of U. Then

$$H_PS_P = H_P(U_PR_P^{-1}) = (U_PR_P^{-1})(R_PB_PR_P^{-1}) + u(b^TR_P^{-1}) = S_P\dot{B}_P + u\dot{b}^T$$

is an equivalent decomposition, in which the matrix S_P is J_P-orthogonal. Next let

$$\dot{u} = \gamma^{-1}(u - S_Pa)$$

be a vector of norm one such that $\dot{u}$ is J_P-orthogonal to the span of U_P, that is, $U_P^TJ_P\dot{u} = 0$. Then the decomposition

$$H_PS_P = S_P(\dot{B}_P + a\dot{b}^T) + \dot{u}(\gamma\dot{b}^T) = S_P\ddot{B}_P + \dot{u}\ddot{b}^T$$

is an equivalent J_P-orthogonal Hamiltonian Krylov-Schur-type decomposition. Finally, let $\dot{S}_P$ be a J_P-orthogonal matrix such that $\ddot{b}^T\dot{S}_P = ||b||_2e_{2k}^T$ and $\dot{S}_P^{-1}\ddot{B}_P\dot{S}_P = \widetilde{H}_P$ is in permuted Hamiltonian J-Hessenberg form (this reduction has to be performed rowwise from bottom to top in order to achieve $\ddot{b}^T\dot{S}_P = ||b||_2e_{2k}^T$, see Table 2.7 for an algorithm which constructs such an $\dot{S}$). Then the equivalent decomposition

$$H_P\ddot{S}_P = H_P(S_P\dot{S}_P) = (S_P\dot{S}_P)(\dot{S}_P^{-1}\ddot{B}_P\dot{S}_P) + \dot{u}(\ddot{b}^T\dot{S}_P) = \ddot{S}_P\widetilde{H}_P + \ddot{u}e_{2k}^T$$

is a possibly reduced symplectic Lanczos factorization. $\surd$

This theorem describes in a constructive way how to pass from a Hamiltonian Krylov-Schur-type sequence to a symplectic Lanczos sequence.

Now let us assume that we have constructed a symplectic Lanczos factorization of order $2(k+p) = 2m$ of the form (8.3)

$$H_P S_P^{2n,2m} = S_P^{2n,2m} \widetilde{H}_P^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T. \tag{8.31}$$

Applying the SR algorithm 5.3 to $\widetilde{H}^{2m,2m}$ and solving all 2×2 and 4×4 subproblems as discussed in Section 6 yields a symplectic matrix $\breve{S}$ such that

$$\breve{S}^{-1} \widetilde{H}^{2m,2m} \breve{S} = \begin{bmatrix} \widetilde{A} & \widetilde{G} \\ \widetilde{Q} & -\widetilde{A}^T \end{bmatrix} = \breve{H}^{2m,2m}$$

decouples into 1×1 or 2×2 blocks on the diagonals of each of the four subblocks $\widetilde{A}, \widetilde{G}$ and $\widetilde{Q}$ as explained in (6.7). Hence $\breve{H}_P^{2m,2m}$ is a block diagonal matrix with 2×2 and 4×4 blocks on the diagonal. Assume furthermore, that $\breve{S}_P$ has been constructed such that the desired eigenvalues of $\widetilde{H}_P^{2m,2m}$ have been moved to the left upper part of $\breve{H}_P^{2m,2m}$, that is the desired eigenvalues are all moved to the $\breve{H}_{11}$ part

$$\breve{H}_P^{2m,2m} = \begin{bmatrix} \breve{H}_{11} & \\ & \breve{H}_{22} \end{bmatrix}.$$

This can easily be achieved by J_P-orthogonal permutation matrices of the form

$$\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$$

as

$$\begin{bmatrix} 0 & I_2 \\ I_1 & 0 \end{bmatrix} \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \begin{bmatrix} 0 & I_1 \\ I_2 & 0 \end{bmatrix} = \begin{bmatrix} B_2 & 0 \\ 0 & B_1 \end{bmatrix}$$

interchanges the diagonal blocks B_1 and B_2. Here the size of the identity matrices I_1, I_2 is the same as that of B_1 and B_2.

Then postmultiplying (8.31) by $\breve{S}_P$

$$H_P S_P^{2n,2m} \breve{S}_P = S_P^{2n,2m} \breve{S}_P \breve{S}_P^{-1} \widetilde{H}_P^{2m,2m} \breve{S}_P + \zeta_{m+1} v_{m+1} e_{2m}^T \breve{S}_P$$

yields a J_P-orthogonal Hamiltonian Krylov-Schur-type decomposition

$$H_P \breve{S}_P^{2n,2m} = \breve{S}_P^{2n,2m} \breve{H}_P^{2m,2m} + \zeta_{m+1} v_{m+1} s_{2m}^T$$

similar to the symplectic Lanczos factorization (8.31). Due to the special form of $\breve{H}_P^{2m,2m}$, the Hamiltonian Krylov-Schur-type decomposition can be partitioned in the form

$$H_P [\breve{S}_P^{2n,2\ell} \ \breve{S}_P^{2n,rest}] = [\breve{S}_P^{2n,2\ell} \ \breve{S}_P^{2n,rest}] \begin{bmatrix} \breve{H}_{11} & \\ & \breve{H}_{22} \end{bmatrix} + \zeta_{m+1} v_{m+1} [\breve{s}_{2\ell}^T \ \breve{s}_{rest}^T]$$

if $\breve{H}_{11} \in \mathbb{R}^{2\ell,2\ell}$. Then

$$H_P \breve{S}_P^{2n,2\ell} = \breve{S}_P^{2n,2\ell} \breve{H}_{11} + \zeta_{m+1} v_{m+1} \breve{s}_{2\ell}^T \tag{8.32}$$

is also a Hamiltonian Krylov-Schur-type decomposition. In other words, a Hamiltonian Krylov-Schur-type decomposition splits at any point where its Rayleigh quotient

is block diagonal. Theorem 8.8 says that there is an equivalent symplectic Lanczos factorization

$$H_P \check{S}_P^{2n,2\ell} = \check{S}_P^{2n,2\ell} \check{H}_P^{2\ell,2\ell} + \check{v}_{2\ell+1} e_{2\ell}^T$$

where $\check{H}_P^{2\ell,2\ell}$ is in permuted Hamiltonian J-Hessenberg form and the columns of $\check{S}_P^{2n,2\ell}$ are J_P-orthogonal. Thus, the purging problem can be solved by applying the permuted SR algorithm to $\widetilde{H}_P^{2k,2k}$, moving the unwanted Ritz values into the $\check{H}_{22}$, truncating the decomposition and returning to a symplectic Lanczos factorization.

The restarting algorithm then is to expand this symplectic Lanczos factorization, compute the Hamiltonian Krylov-Schur-type decomposition, move the desired eigenvalues to the beginning, throw away the rest of the decomposition and transform the decomposition back to a symplectic Lanczos one. The symplectic Lanczos factorization achieved in this way is equivalent to the one the implicitly restarted symplectic Lanczos algorithm would achieve if the same Ritz values are discarded in both (and those Ritz values are distinct from the other Ritz values). The proof follows the lines of the proof of Theorem 3.1 in [130] or Theorem 2.4 of Chapter 5 in [131].

As the iteration progresses, the Ritz estimates will converge at different rates. When a Ritz estimate is small enough, the corresponding Ritz value is said to have converged. The converged Ritz value may be wanted or unwanted. Unwanted ones can be deflated from the current factorization using the above procedure. Wanted ones should be deflated in the following sense to speed up convergence.

Assume that we have achieved a Hamiltonian Krylov-Schur-type decomposition (8.32)

$$H_P[\check{S}_P^1 \ \check{S}_P^2] = [\check{S}_P^1 \ \check{S}_P^2] \begin{bmatrix} \check{H}_1 & \\ & \check{H}_2 \end{bmatrix} + \check{v}_{m+1}[0 \ s_2^T] \tag{8.33}$$

where $\check{H}_2 \in \mathbb{R}^{2j\times 2j}, \check{S}_P^2 \in \mathbb{R}^{2n,2j}, s_2 \in \mathbb{R}^{2j}$. That is, we have $H_P\check{S}_P^1 = S_P^1\check{H}_1$, so that $\check{S}_P^1$ spans an eigenspace of H_P. We say a Hamiltonian Krylov-Schur-type decomposition has been deflated if it can be partitioned in this form. After deflation, equating the last $2j$ columns of (8.33) results in

$$H_P\check{S}_P^2 = \check{S}_P^2\check{H}_2 + \check{v}_{m+1}s_2^T.$$

As Stewart [131, 130] points out, there are two advantages to deflating converged eigenspaces. First, by freezing it at the beginning of the Hamiltonian Krylov-Schur-type decomposition, we insure that the remaining space of the decomposition remains J_P-orthogonal to it. In particular, this gives algorithms the opportunity to compute more than one independent eigenvector corresponding to a multiple eigenvalue.

The second advantage of the deflated decomposition is that we can save operations in the contraction phase of the Krylov-Schur-type cycle. Only the rightmost part of the Hamiltonian Krylov-Schur-type decomposition will be transformed back to a symplectic Lanczos factorization

$$H_P[\check{S}_P^1 \ \dot{S}_P^2] = [\check{S}_P^1 \ \dot{S}_P^2] \begin{bmatrix} \check{H}_1 & \\ & \dot{H}_2 \end{bmatrix} + \dot{v}_{m+1}e_{2\ell}^T.$$

The expansion phase does not change, and we end up with a decomposition of the form

$$H_P[\check{S}_P^1 \ \dot{S}_P^2 \ \dot{S}_P^3] = [\check{S}_P^1 \ \dot{S}_P^2 \ \dot{S}_P^3] \begin{bmatrix} \check{H}_1 & & \\ & \dot{H}_2 & \dot{H}_{23} \\ & \dot{H}_{32} & \dot{H}_{33} \end{bmatrix} + \check{v}_{m+1}e_{2\ell}^T.$$

Since $\breve{H}_1$ is uncoupled from the rest of the Rayleigh quotient, we can apply all subsequent transformations exclusively to the eastern part of the Rayleigh quotient and to $[\dot{S}_P^2\ \dot{S}_P^3]$. If the order of $\breve{H}_1$ is small, the savings will be marginal; but as its size increases during the course of the algorithm, the savings become significant.

While the implicitly restarted symplectic Lanczos factorization (8.22) can restart with an arbitrary filter polynomial, the Krylov-Schur-type method discussed here cannot do that. When it comes to exact shifts the Krylov-Schur-type method is to be preferred because exchanging eigenvalues in a Schur-type form is a more reliable process than using implicit SR steps to deflate.

8.6. Stability Issues. There are different source of instability in the (implicitly restarted) symplectic Lanczos process: the use of the potentially unstable SR algorithm, Lanczos vectors with norm not equal to one and the loss of J-orthogonality between the symplectic Lanczos vectors. In the following, we will briefly address these issues.

It is well known that for general Lanczos–like methods the stability of the overall process is improved when the norm of the Lanczos vectors is chosen to be equal to 1 [112, 133]. Thus, Freund and Mehrmann propose in [64] to modify the prerequisite $S_P^T J_P S_P = J_P$ of the symplectic Lanczos method to

$$S_P^T J_P S_P = \operatorname{diag}(\begin{bmatrix} 0 & \sigma_1 \\ -\sigma_1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & \sigma_2 \\ -\sigma_2 & 0 \end{bmatrix}, \ldots, \begin{bmatrix} 0 & \sigma_n \\ -\sigma_n & 0 \end{bmatrix}) =: \Sigma$$

and

$$||v_j||_2 = ||w_j||_2 = 1, \qquad j = 1, \ldots, n.$$

For the resulting algorithm and a discussion of it we refer to [64]. It is easy to see that $\widetilde{H}_P = S_P^{-1} H_P S_P$ is no longer a permuted Hamiltonian J–Hessenberg matrix, as S is only 'almost' symplectic, but

$$\Sigma \widetilde{H}_P = (\Sigma \widetilde{H}_P)^T.$$

Thus $\widetilde{H} = P^T \widetilde{H}_P P$ still has the desired form of a Hamiltonian J–Hessenberg matrix but the upper right $n \times n$ block is no longer symmetric. Therefore $\widetilde{H}$ is diagonally similar to a Hamiltonian J–Hessenberg matrix.

Unfortunately an SR step does not preserve this structure and thus this modified version of the symplectic Lanczos method cannot be used in connection with our restart approaches.

One important property for a stable implicitly restarted Lanczos method is that the Lanczos vectors stay bounded after possibly many implicit restarts. Neither for the symplectic Lanczos method nor for the symplectic SR algorithm it can be proved that the symplectic transformation matrix stays bounded. Hence the symplectic Lanczos vectors $S_P^{2n,2k}$ computed via an implicitly restarted symplectic Lanczos method may not stay bounded; this has to be monitored during the iteration. During the SR step on the $2k \times 2k$ symplectic butterfly matrix, all but $k-1$ transformations are orthogonal. These are known to be numerically stable. For the $k-1$ nonorthogonal symplectic transformations that have to be used, we choose among all possible transformations the ones with optimal (smallest possible) condition number (see [38]).

8.6.1. Re-J-Orthogonalization. In theory, the above recurrences for v_m and w_m are sufficient to guarantee the J–orthogonality of theses vectors. Yet, in practice, the J–orthogonality will be lost, and some form of reorthogonalization any Lanczos algorithm is numerically unstable. Hence we re–J–orthogonalize each Lanczos vector as soon as it is computed against the previous ones via

$$\begin{aligned} w_m &= w_m + S_P^{2n,2m-2} J_P^{2(m-1),2(m-1)} S_P^{2n,2m-2^T} J_P^{2n,2n} w_m, \\ v_{m+1} &= v_{m+1} + S_P^{2n,2m} J_P^{2m,2m} S_P^{2n,2m^T} J_P^{2n,2n} v_{m+1}. \end{aligned}$$

A different way to write this re–J_P–orthogonalization is

$$\begin{aligned} w_m &\leftarrow w_m + \sum_{j=1}^{m-1} \left(< v_j, w_m >_{J_P} w_j - < w_j, w_m >_{J_P} v_j \right), \\ v_{m+1} &\leftarrow v_{m+1} + \sum_{j=1}^{m} \left(< v_j, v_{m+1} >_{J_P} w_j - < w_j, v_{m+1} >_{J_P} v_j \right), \end{aligned}$$

where for $x, y \in \mathbb{R}^{2n}$, $< x, y >_{J_P} := x^T J_P^{2n,2n} y$ defines the indefinite inner product implied by $J_P^{2n,2n}$.

This re–J–orthogonalization is costly, it requires $16n(m-1)$ flops for the vector w_m and $16nm$ flops for v_{m+1}. Thus, if $2k$ Lanczos vectors $v_1, w_1, \ldots, v_k, w_k$ are computed, the re–J–orthogonalization adds $16n(k+1)k - 32n$ flops to the overall cost of the symplectic Lanczos method.

For standard Lanczos algorithms, different reorthogonalization techniques have been studied (for references see, e.g., [65]). Those ideas can be used to design analogous re–J–orthogonalizations for the symplectic Lanczos method.

8.7. Implicit versus Explicit Restarts. Implicit restarts have some advantages over explicit restarts as will be discussed in this section. First of all, implicit restarts are more economical to implement. Assume we have to employ a restart after k steps of the symplectic Lanczos method. An implicit single shift restart requires

	$28n \cdot k + 16n + (100k - 65)$	flops for the implicit SR step
and	$38n + 4nz$	flops for one additional Lanczos step
and	$32n \cdot k - 16n$	flops for re–J–orthogonalization.

That is a total of $4nz + 60n \cdot k + 38n + 100k - 65$ flops.
An explicit restart requires

	$4nz \cdot k + 32n \cdot k + 6n$	flops for k Lanczos steps
and	$16n \cdot (k+1)k - 32n$	flops for re–J–orthogonalization.

This sums up to $4nz \cdot k + 16n \cdot k^2 + 48n \cdot k - 26n$ flops. If an explicit restart with the starting vector $\check{v}_1 = (H_P - \mu I)v_1$ would be performed, this would add another $8n^2 + 2n$ to this flop count.

From these numbers we can conclude that performing an implicit restart is significantly cheaper than explicitly restarting the Lanczos iteration. This is due to the fact that an implicit SR step is usually cheaper than k Lanczos steps. Besides we have to re–J–orthogonalize only once while an explicit restart would require a re–J–orthogonalization in each iteration step. For more economical re–J–orthogonalization techniques implicit restarts are also advantageous. For double–shifted or multishifted

restarts the implicit technique is still favorable although the difference in the flop count becomes smaller.

Performing an explicit restart with $(H_P - \mu I)v_1$ or $(H_P - \mu I)(H_P + \mu I)v_1$ or $(H_P - \mu I)(H_P + \mu I)(H_P - \overline{\mu} I)(H_P + \overline{\mu} I)$ as the new starting vector, one is forced to directly multiply the old starting vector by matrices of the form $(H_P - xI)$. This can be avoided by the implicit method.

Note that the starting vector v_1 can be expressed as a linear combination of the eigenvectors y_i of H_P (assuming for simplicity that H_P is diagonalizable) :

$$v_1 = \sum_{i=1}^{2n} \alpha_i y_i.$$

Then a single shifted starting vector takes the form

$$\breve{v}_1 = \rho(H_P - \mu I)v_1 = \rho \sum_{i=1}^{2n} \alpha_i(\lambda_i - \mu)y_i$$

where the λ_i are the eigenvalues corresponding to y_i. As the single shift selected will be real, applying such a modification to v_1 tends to emphasize those eigenvalues of H_P in $\breve{v}_1$ which correspond to eigenvalues λ_i with the largest positive or negative real part (depending on whether the chosen shift is positive or negative). Thus it is possible that the vector $\breve{v}_1$ will be dominated by information only from a few of the eigenvalues with largest real part. An implicit restart directly forms $\breve{S}_P^{2k}$ from a wide range of information available in S_P^{2k} and this should give better numerical results than the explicit computation of $\breve{v}_1$.

As an example consider

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

where $A = \mathrm{diag}(-10^6, 9, 8, 7, 6, 5, 4, 3, \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix})$ is a block diagonal matrix and U is the product of randomly generated symplectic Householder and Givens matrices. The eigenvalues of H can be read off directly. The following computations were done using MATLAB. The starting vector v_1 is chosen randomly. After 4 steps of the symplectic Lanczos method the resulting 8×8 Hamiltonian J–Hessenberg matrix $\widetilde{H}^8$ has the eigenvalues (computed by the MATLAB function `eig`)

$$\lambda(\widetilde{H}^8) = \left\{ \begin{array}{r} 9.999999999999997e+05 \\ -9.999999999999997e+05 \\ 3.040728370123861e+00 \\ -3.040728370123995e+00 \\ 9.200627380564711e+00 \\ -9.200627380564642e+00 \\ 9.477682371618508e+00 \\ -9.477682371618551e+00 \end{array} \right\}.$$

To remove an eigenvalue pair from $\widetilde{H}^8$ one can perform an implicitly double–shifted restart described in Section 8.4. Removing the two eigenvalues of smallest absolute

value from $\widetilde{H}^8$, we obtain a Hamiltonian J–Hessenberg matrix $\breve{H}^6_{impl}$ whose eigenvalues are

$$\lambda(\breve{H}^6_{impl}) = \left\{ \begin{array}{r} 9.999999999999994e+05 \\ -9.999999999999994e+05 \\ 9.200627382497721e+00 \\ -9.200627382497721e+00 \\ 9.477682372414739e+00 \\ -9.477682372414737e+00 \end{array} \right\}.$$

From (4.2) it follows that these have to be the 6 eigenvalues of $\widetilde{H}^8$ which have not been removed. As can be seen, we loose $4-5$ digits during the implicit restart. Performing an explicit restart with the explicitly computed new starting vector $\breve{v}_1 = (H-\mu I)(H+\mu I)v_1$ yields a Hamiltonian J–Hessenberg matrix $\breve{H}^6_{expl}$ with eigenvalues

$$\lambda(\breve{H}^6_{expl}) = \left\{ \begin{array}{r} 9.999999999999999e+05 \\ -9.999999999999999e+05 \\ 9.200679454660859e+00 \\ -9.200679454660861e+00 \\ 9.477559041923007e+00 \\ -9.477559041923007e+00 \end{array} \right\}.$$

This time we lost up to 10 digits. As a general observation from a wide range of numerical tests, the explicit restart looses at least 2 digits more than the implicit restart.

8.8. Breakdowns in the SR Factorization. So far we have assumed that the SR decomposition always exists. Unfortunately this assumption does not always hold. If there is a starting vector $\breve{v}_1 = q(H)v_1$ for which the explicitly restarted symplectic Lanczos method breaks down, then it is impossible to reduce the Hamiltonian matrix H to Hamiltonian J–Hessenberg form with a transformation matrix whose first column is $\breve{v}_1$. Thus, in this situation the SR decomposition of $q(H)$ cannot exist.

As will be shown in this section, this is the only way that breakdowns in the SR decomposition can occur. For simplicity, we will discuss the single shift SR step (that is, a spectral function $q(H_P) = H_P - \mu I$), in which only two types of elementary transformations are used. Most of them are orthogonal symplectic Givens rotations, their computation cannot break down. The other type of transformations used are symplectic Gaussian eliminations. These are the only possible source of breakdown.

Assuming that using the symplectic Lanczos we have achieved (8.3),

$$H_P S_P^{2n,2k} = S_P^{2n,2k} \widetilde{H}_P^{2k,2k} + \widetilde{\zeta}_{k+1} v_{k+1} e_{2k}^T, \tag{8.34}$$

where $S_P^{2n,2k} = [v_1, w_1, \ldots, v_k, w_k]$ and the parameters of $\widetilde{H}_P^{2k,2k}$ are given by $\widetilde{\delta}_1, \ldots, \widetilde{\delta}_k$, $\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k$, $\widetilde{\nu}_1, \ldots, \widetilde{\nu}_k$, and $\widetilde{\zeta}_2, \ldots, \widetilde{\zeta}_k$. Next an implicit single shift restart is performed. The single shift implicit SR step applies a sequence of Givens and Gauss transformations. Assume that the first $j-1$ Gauss transformation exist. Then the first part of the transformation of $\widetilde{H}^{2k,2k}$ can be computed

$$G_j L_j \cdots G_2 L_2 G_1 =: \widehat{S}^{-1}$$

as described in Section 5.1 such that the first $j-1$ columns and the columns $k+1, \ldots, k+j-1$ of

$$\widehat{S}^{-1} \widetilde{H}^{2k,2k} \widehat{S}$$

are in J-Hessenberg form. The bulge has been moved to the $(j+2, j+1)$ position (see Section 5.1). In order to use the same notation as in the rest of the discussion on the symplectic Lanczos method, let us consider the permuted version

$$\widehat{S}_P^{-1} \widetilde{H}_P^{2k,2k} \widehat{S}_P = \breve{H}_P^{2k,2k}.$$

Moreover, as only the first j steps of the SR decomposition are performed, we have

$$\widehat{S}_P = \begin{bmatrix} \widehat{S}_P^{2j,2j} & 0 \\ 0 & I \end{bmatrix},$$

and $\widehat{S}_P^{2j,2j}$ is an upper Hessenberg matrix.

Consider the leading $(2j+2)\times(2j+2)$ principal submatrix of $\breve{H}_P^{2k,2k}$. The leading $2j \times 2j$ principal submatrix is in permuted J-Hessenberg form, while the last two rows and columns have the following form

$$\breve{H}_P^{2j+2,2j+2} = \left[\begin{array}{cc|cc|cc|cc} \breve{\delta}_1 & \breve{\beta}_1 & 0 & \breve{\zeta}_2 & & & & \\ \breve{\nu}_1 & -\breve{\delta}_1 & 0 & 0 & & & & \\ \hline 0 & \breve{\zeta}_2 & \ddots & & \ddots & & & \\ 0 & 0 & & \ddots & & \ddots & & \\ \hline & & \ddots & & \breve{\delta}_j & \overline{\beta_j} & 0 & x_2 \\ & & & \ddots & \overline{\nu}_j & -\breve{\delta}_j & 0 & -x_1 \\ \hline & & & & x_1 & x_2 & x & x \\ & & & & 0 & 0 & x & x \end{array}\right]$$

as $\widetilde{\zeta}_{j+1} e_{2j}^T \widehat{S}_P^{2j,2j} = [0, \cdots, 0, x_1, x_2]^T$ because $\widehat{S}_P^{2j,2j}$ is an upper Hessenberg matrix. The parameters $\overline{\beta}_j$ and $\overline{\nu}_j$, as well as the entries of the trailing 2×2 principal submatrix, are not in their form yet. In the implicit SR step a Gauss transformation would be used next to eliminate x_1 against $\overline{\nu}_j$. The elimination will fail to exist if $\overline{\nu}_j = 0$; in that case the SR decomposition of $q(\widetilde{H}^{2k,2k})$ does not exist.

Next, we will show that this breakdown in the SR decomposition implies a breakdown in the Lanczos process started with the starting vector $\breve{v}_1 = \rho q(H_P) v_1$.

From (8.34) we obtain by multiplying from the right by $\widehat{S}_P$

$$H_P \breve{S}_P^{2n,2k} = \breve{S}_P^{2n,2k} \breve{H}_P^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T \widehat{S}_P$$

where $\breve{S}_P^{2n,2k} = S_P^{2n,2k} \widehat{S}_P = [\breve{v}_1, \breve{w}_1, \ldots, \breve{v}_j, \breve{w}_j, v_{j+1}, w_{j+1}, \ldots, v_k, w_k]$. From the derivations in the discussion of the implicit restart, we know that the starting vector of this recursion is given by $\breve{v}_1 = \rho q(H_P) v_1$. As the trailing principal submatrix of $\widehat{S}_P$ is the identity, we can just as well consider

$$H_P \breve{S}_P^{2n,2j} = \breve{S}_P^{2n,2j} \breve{H}_P^{2j,2j} + r_j e_{2j}^T$$

for some r_j. $\breve{H}_P^{2j,2j}$ is in Hamiltonian J-Hessenberg form. Due to the special form of $\breve{H}_P^{2j+1,2j+2}$, the next step in the symplectic Lanczos process determines the vector $\breve{w}_j$ using the equation

$$H_P \breve{v}_j = \breve{\delta}_j \breve{v}_j + \overline{\nu}_j \breve{w}_j + x_1 v_{j+1}. \tag{8.35}$$

The vectors $\breve{v}_j$ and v_{j+1} are known, as well as the scalars $\breve{\delta}_j$ and x_1. Hence,

$$\breve{w}_j = \frac{1}{\overline{\nu}_j}(-H_P\breve{v}_j + \breve{\delta}_j\breve{v}_j + x_1 v_{j+1}).$$

Premultiplying (8.35) by $\breve{v}_j^T J_P$ we obtain

$$\breve{v}_j^T J_P H_P \breve{v}_j = \breve{\delta}_j \underbrace{\breve{v}_j^T J_P \breve{v}_j}_{=0} + \overline{\nu}_j \underbrace{\breve{v}_j^T J_P \breve{w}_j}_{=1} + x_1 \underbrace{\breve{v}_j^T J_P v_{j+1}}_{=0} = \overline{\nu}_j$$

as the columns of $\breve{S}_p^{2n,2k}$ are J_P-orthogonal. The symplectic Lanczos will break down if $\overline{\nu}_j = \breve{v}_j^T J_P H_P \breve{v}_j = 0$, that is, an SR breakdown implies a serious Lanczos breakdown. The opposite implication follows from the uniqueness of the SR decomposition and of the Lanczos factorization.

Hence, we have

THEOREM 8.9. *Suppose the Hamiltonian J–Hessenberg matrix $\widetilde{H}^{2k}$ corresponding to (8.3) is unreduced and let $\mu \in \mathbb{R}$. Let $G_P(j,y)$ be the jth permuted symplectic Gauss transformation required in the SR step on $(\widetilde{H}_P^{2k} - \mu I)$. If the first $j-1$ permuted symplectic Gauss transformations of this SR step exist, then $G_P(j,y)$ fails to exist if and only if $\breve{v}_j^T J_P H_P \breve{v}_j = 0$ with $\breve{v}_j$ as in (8.24).*

Similar theorems can be shown for the double and quadruple shift case. A similar statement for the nonsymmetric Lanczos method and the HR decomposition has been given in [66, Theorem 3].

9. Numerical Experiments.

9.1. Passivity preserving model reduction via a structured Lanczos method. This section is concerned with linear time invariant (LTI) systems

$$\Sigma: \quad \dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t) \tag{9.1}$$

where $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times p}, C \in \mathbb{R}^{p\times n}, D \in \mathbb{R}^{p\times p}$. In this setting, u is the input or excitation function, x is the state, and the function $f(x,u) = Ax(t)+Bu(t)$ determines the dynamics of the system Σ. y is the output or set of observations and $h(x,u) = Cx(t)+Du(t)$ describes the way that the observations are deduces from the state and the input. The complexity of Σ is defined as the number of states n. The problem we will address is to approximate Σ with another dynamical system

$$\widehat{\Sigma}: \quad \dot{\hat{x}}(t) = \widehat{A}\hat{x}(t) + \widehat{B}u(t), \quad \hat{y}(t) = \widehat{C}\hat{x}(t) + \widehat{D}u(t), \tag{9.2}$$

where $\widehat{A} \in \mathbb{R}^{\ell\times\ell}, \widehat{B} \in \mathbb{R}^{\ell\times p}, \widehat{C} \in \mathbb{R}^{p\times\ell}, \widehat{D} \in \mathbb{R}^{p\times p}$. That is, in the new system the number of states (the number of first order differential equations to be solved) is much less than in the original system: $\ell \ll n$. System properties of the original system such as stability, passivity, controllability or observability should be preserved by a model reduction procedure. Often, the existence of a global error bound is required and a small approximation error in terms of $||y - \widehat{y}||$ for an appropriate norm is desired.

Most approaches for model reduction of LTI systems are based on the idea of projecting the original system Σ onto a system of lower order. Usually, two real $n \times \ell$ matrices V and W with $W^T V = I$ are computed which define the projector $\Pi = VW^T$. The projection of the states of the original system generates a reduced-order model:

$$\widehat{A} = W^T AV, \quad \widehat{B} = W^T B, \quad \widehat{C} = CV, \quad \widehat{D} = D. \tag{9.3}$$

For a recent survey of methods for computing reduced-order models see [5, 25].

Here we will consider only stable and passive systems Σ. A system is stable, if the matrix A is stable, that is, if all eigenvalues of A lie in the open left half plane. A system is passive if it does not generate any energy internally, and strictly passive, if it consumes or dissipates input energy. A classical result [144] says that a passive system is positive real. For the LTI systems considered here this implies that the corresponding transfer function $G(s) = D + C(sI - A)^{-1}B$ is positive real, that is, $G(s)$ is analytic and $G(s) + (G(s))^H \geq 0$ for $\mathrm{Re}(s) > 0$ (see, e.g., [10] and the references therein). When reducing a stable and passive system, it is important to produce a reduced-order model that preserves the important system properties and response characteristics such as stability and passivity while at the same time using a computationally efficient method.

Passive system arise in a variety of control problems for mechanical, mechatronic and micro-electro-mechanical systems, see [78]. Circuit simulation is another important source of model reduction problems. In some of these applications, e.g., in the simulation of RLC circuits, the system Σ is stable and passive. A number of methods previously proposed for solving the problem considered here is based on the moment matching property of Krylov subspace methods. For a recent survey of such methods see [62].

Based on recent work by Antoulas [6] characterizing passivity through interpolation conditions, Sorensen derives in [128] a novel projection method that preserves both stability and passivity. Given 2ℓ distinct points $s_1, \ldots, s_{2\ell}$, let $T_j = s_j I - A$ and

$$\widetilde{V} = [T_1^{-1}B \ \ldots \ T_\ell^{-1}B], \tag{9.4}$$

$$\widetilde{W} = [T_{k+1}^{-T}C^T \ \ldots \ T_{2\ell}^{-T}C^T]. \tag{9.5}$$

Assuming that $\det \widetilde{W}^T\widetilde{V} \neq 0$, the projected system defined by (9.3) where $V = \widetilde{V}$ and $W = \widetilde{W}(\widetilde{V}^T\widetilde{W})^{-1}$, interpolates the transfer function of Σ at the points s_i (see [6, Proposition 4.1]):

$$\widehat{G}(s_i) = G(s_i), \quad i = 1, 2, \ldots, 2\ell.$$

Antoulas proves in [6], that if the interpolation points s_j in (9.4), (9.5) are chosen as spectral zeros of the original passive system Σ, the reduced system $\widehat{\Sigma}$ defined by (9.3) is both stable and passive. Note that in order to end up with real reduced matrices $\widehat{A}, \widehat{B}, \widehat{C}$, whenever a complex spectral zero is selected, its complex conjugate has to be selected as well. The matrices $\widetilde{V}$, $\widetilde{W}$ can be obtained without the explicit computation of spectral zeros. As Sorensen [128] noted, this can be achieved through the computation of certain invariant subspaces of a generalized eigenvalue problem. In the following, we will briefly review the results by Antoulas [6] and Sorensen [128] and state the passivity preserving model reduction method proposed by Sorensen. We show how the approach can be rewritten such that as a structured eigenvalue problem results and describe an efficient, structure-preserving Krylov subspace method for its solution. Finally, a numerical example comparing Sorensen's method with the structure-preserving one proposed here is reported.

9.1.1. Sorensen's passivity preserving model reduction method. Let us consider the LTI system (9.1) where it is assumed that

1. A is stable,
2. Σ is observable and controllable,

3. $D_+ = D + D^T$ is symmetric positive definite,
4. Σ is passive.

Recall, that a real rational function $G(s)$ is positive real if $G(s)$ is analytic and $G(s) + G^T(-s) \geq 0$ for $Re(s) > 0$. The last property implies here the existence of a stable rational matrix function $W(s)$ such that $G(s) + G^T(-s) = W(s)W^T(-s)$. This is the spectral factorization of G, W is called a spectral factor of G and the zeros of W, i.e., $\lambda_i, i = 1, \ldots, n$, such that $\det W(\lambda_i) = 0$, are called the spectral zeros of G; these definitions and relations can be found in many papers and texts, see e.g. [146, 10]. Denote the set of all spectral zeros by $\mathcal{S}_G$

$$\mathcal{S}_G := \{\lambda | \det\ W(\lambda) = 0\} = \ \{\text{spectral zeros of } G\}.$$

Suppose a reduced-order model $\widehat{\Sigma}$ (9.2) has been obtained and let $\widehat{G}(s) = \widehat{D} + \widehat{C}(sI - \widehat{A})^{-1}\widehat{B}$ be the associated transfer function. Antoulas notes in [6] that a passive reduced-order model $\widehat{\Sigma}$ will result if certain of the spectral zeros are preserved in the reduced-order model.

Proposition: *If $\mathcal{S}_{\widehat{G}} \subset \mathcal{S}_G$, $\widehat{G}(\lambda) = G(\lambda)$ for all $\lambda \in \mathcal{S}_{\widehat{G}}$, and $\widehat{G}$ is a minimal degree rational interpolant of the values of G on the set $\mathcal{S}_{\widehat{G}}$, then $\widehat{\Sigma}$ is both stable and passive.*

The set $\mathcal{S}_G$ of all spectral zeros is equal to the set of (finite) eigenvalues of

$$\mathcal{A} - \lambda\mathcal{E} = \begin{bmatrix} A & & B \\ & -A^T & -C^T \\ C & B^T & D_+ \end{bmatrix} - \lambda \begin{bmatrix} I & & \\ & I & \\ & & 0 \end{bmatrix}, \tag{9.6}$$

that is $\mathcal{S}_G = \sigma(\mathcal{A}, \mathcal{E}) \backslash \{\infty\}$ where

$$\sigma(\mathcal{A}, \mathcal{E}) = \{\lambda \in \mathbb{C} | \det(\mathcal{A} - \lambda\mathcal{E}) = 0\}.$$

Making use of this observation, Sorensen suggests in [128] to compute the reduced-order model through the construction of a basis for a selected invariant subspace of the pair $(\mathcal{A}, \mathcal{E})$. That is, he suggests to compute a partial real generalized Schur decomposition of $(\mathcal{A}, \mathcal{E})$ such that

$$U^T\mathcal{A}Q = S$$

is an upper quasi-triangular matrix in $\mathbb{R}^{\ell \times \ell}$ and

$$U^T\mathcal{E}Q = T$$

is an upper triangular matrix in $\mathbb{R}^{\ell \times \ell}$ and all eigenvalues of (S, T) have positive real part. The matrices U and Q in $\mathbb{R}^{(2n+p) \times \ell}$ have orthonormal columns. Then T is nonsingular and we have from $\mathcal{A}Q = US, \mathcal{E}Q = UT$ that $\mathcal{E}QT^{-1} = U$ and

$$\mathcal{A}Q = \mathcal{E}QT^{-1}S =: \mathcal{E}QR$$

where Q has orthononormal columns ($Q^TQ = I$) and R is real quasi-upper triangular in $\mathbb{R}^{\ell \times \ell}$ such that $\mathrm{Re}(\lambda) > 0$ for all eigenvalues λ of R.

Let $Q^T = [X^T, Y^T, Z^T]$ be partitioned in accordance with the block structure of $\mathcal{A}$; $X \in \mathbb{R}^{n \times \ell}, Y \in \mathbb{R}^{n \times \ell}, Z \in \mathbb{R}^{p \times \ell}$. Then

$$\begin{bmatrix} A & & B \\ & -A^T & -C^T \\ C & B^T & D_+ \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X \\ Y \\ 0 \end{bmatrix} R. \tag{9.7}$$

The desired projection matrices V and W can be computed from X and Y in the following way: Compute the singular value decomposition [65] of

$$X^T Y = Q_x S^2 Q_y^T,$$

where Q_x and Q_y are orthogonal matrices in $\mathbb{R}^{\ell\times\ell}$ and $S^2 \in \mathbb{R}^{\ell\times\ell}$ is a diagonal matrix with nonnegative diagonal entries. Set

$$V = XQ_x S^{-1}, \quad W = YQ_y S^{-1},$$

and

$$\widehat{A} = W^T AV, \quad \widehat{B} = W^T B, \quad \widehat{C} = CV, \quad \widehat{D} = D.$$

The resulting reduced system is stable and passive [128].

For small to medium size dense problems $\mathcal{A}$ and $\mathcal{E}$ can be actually formed and the desired generalized Schur decomposition can be obtained from the full one. For large sparse systems this would be impractical and inefficient. An iterative method computing a desired set of eigenvalues and associated eigenvectors (or an associated invariant subspace) is more appropriate. As the best currently available method for this purpose, the implicitly restarted Arnoldi (IRA) algorithm [126] as implemented in MATLAB's eigs or ARPACK in Fortran [90] can not deal with the problem under consideration here (as $\mathcal{E}$ is not positive definite), Sorensen suggests applying a Cayley transformation $\mathcal{C}_\mu = (\mu\mathcal{E} - \mathcal{A})^{-1}(\mu\mathcal{E} + \mathcal{A})$ where $\mu \geq 0$ is a real shift. With a proper choice of μ this will provide for rapid convergence to an invariant subspace corresponding to the ℓ transformed eigenvalues of largest magnitude of

$$(\mu\mathcal{E} - \mathcal{A})^{-1}(\mu\mathcal{E} + \mathcal{A})Q = Q\widehat{R} \tag{9.8}$$

so that

$$\mathcal{A}Q = \mathcal{E}QR \quad \text{where} \quad R := \mu(\widehat{R} - I)(\widehat{R} + I)^{-1}. \tag{9.9}$$

An implementation will require two sparse direct factorizations of $A - \mu I$ and $A + \mu I$. The Cayley transformation may then be applied to an arbitrary vector using a blocked matrix-vector product followed by a Gaussian block elimination. Moreover, it should be noted that as the partial real Schur decomposition will automatically keep complex conjugate pairs of eigenvalues together, the parameter ℓ that specifies the order of the reduced system will perhaps need to be adjusted by 1 to accommodate this. The algorithm as posed is appropriate for real matrices, and in particular, all arithmetic stays real throughout. Please note the use of MATLAB's `'eig'` will introduce complex arithmetic as eigenvectors and not invariant subspaces will be computed. This complex data has to be transformed to equivalent real one in order to be used in the above process.

9.1.2. Hamiltonian eigenproblem. It is easily seen that $\lambda \in \mathcal{S}_G$ implies $-\overline{\lambda} \in \mathcal{S}_G$ since from $\mathcal{A}u = \lambda\mathcal{E}u$ we have $\widetilde{u}^T\mathcal{A} = -\overline{\lambda}\widetilde{u}^T\mathcal{E}$, where $u^T = [x^T, y^T, z^T]$ and $\widetilde{u}^T = [y^T, -x^T, z^T]$. As Hamiltonian matrices display such an eigenvalue pairing, in this section we will show how to transform the generalized eigenvalue problem $\mathcal{A} - \lambda\mathcal{E}$ associated with the generalized Schur decomposition (9.7) into a standard eigenvalue problem for a Hamiltonian matrix.

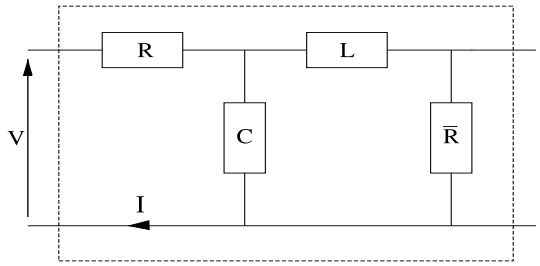


FIG. 9.1. *one section of the circuit as in [67]*

From (9.7) we obtain [5]

$$AX + BZ = XR \tag{9.10}$$
$$-A^T Y - C^T Z = YR \tag{9.11}$$
$$CX + B^T Y + D_+ Z = 0 \tag{9.12}$$

From (9.12) it follows that $Z = -D_+^{-1}\{CX + B^T Y\}$, as $D_+ = D + D^T$ is symmetric positive definite (assumption 3 in Section 9.1.1). Substituting this expression for Z into (9.10), (9.11) yields

$$\underbrace{\begin{bmatrix} A - BD_+^{-1}C & -BD_+^{-1}B^T \\ C^T D_+^{-1}C & -(A - CD_+^{-1}B)^T \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} R.$$

$\mathcal{H}$ can be represented as

$$\mathcal{H} = \begin{bmatrix} \widetilde{A} & \widetilde{G} \\ \widetilde{Q} & -\widetilde{A}^T \end{bmatrix} \tag{9.13}$$

with $\widetilde{G} = \widetilde{G}^T$ and $\widetilde{Q} = \widetilde{Q}^T$. Obviously, $\mathcal{H}$ is a Hamiltonian matrix and its eigenvalues are eigenvalues of the pair $(\mathcal{A}, \mathcal{E})$. Hence, instead of considering the partial generalized Schur decomposition

$$\mathcal{A}Q = \mathcal{E}QR$$

we can consider the partial Schur decomposition of the Hamiltonian matrix $\mathcal{H}$

$$\mathcal{H}S = S\Lambda, \tag{9.14}$$

where $S \in \mathbb{R}^{2n\times\ell}$ and $\Lambda \in \mathbb{R}^{\ell\times\ell}$.

Using the symplectic Lanczos algorithm for solving this eigenproblem, we not only save about half of the computational effort compared to the standard approach; this approach offers a fast, efficient and more reliable computation of the reduced-order model. Unfortunately, as for all model reduction methods based on Krylov subspace methods, there does not exist a global error bound.

9.1.3. Numerical Experiment. For the preliminary numerical experiment presented in this section, we consider the RLC ladder network of [67]. This circuit consists of 200 sections interconnected in cascade; each section is as shown in Fig. 9.1. The

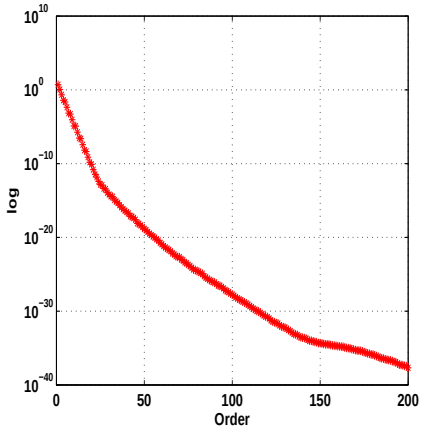


FIG. 9.2. *Hankel singular values*

input is the voltage V applied to the first section; the output is the current I of the first section. The order of the overall system is $n = 400$. The state variable x_{2i-1} is the voltage across capacitor C_i, for $i = 1, 2, \ldots, \frac{n+1}{2}$, while the state variable x_{2i} is the current through the inductor L_i for $i = 1, 2, \ldots, \frac{n-1}{2}$. It is assumed that all the capacitors, inductors and resistors have the value 0.1, except $\overline{R} = 1$. The Hankel singular vales of this system decay rapidly (see Figure 9.2) so that one can expect a good approximation of the original system by a reduced-order one.

We compare the two approaches described before. For the first approach the matrix pencil $\mathcal{A} - \lambda\mathcal{E}$ (9.6) is set up. A shift μ is chosen (here $\mu = 5$), the Cayley transformation (9.8) is computed (actually, not the matrix itself, but an operator that applies this matrix to a vector is set up) and MATLAB's `'eigs'` is used to compute a $\ell \times \ell$ matrix R and the associated matrix Q as in (9.9). Our implementation makes sure that the computed Q is turned into a real matrix which spans the same subspace os Q. From this the reduced-order system is computed as described in Section 9.1.1. This approach will be referred to in the sequel as the Cayley approach.

The invariant subspace computed determines the choice of the spectral zeros to be interpolated. Fig. 9.3 displays the spectral zeros of the transfer function of the original system in the right half plane (please note that an identical set of spectral zeros can be found in the left half plane) and the spectral zeros chosen by the Cayley approach when $\ell = 10$ is chosen.

The second approach works with the Hamiltonian matrix $\mathcal{H}$ (9.13). A desired set of eigenvalues and appropriate invariant subspaces (9.14) are computed via the structure-preserving symplectic Lanczos algorithm and the reduced system is computed as described in Section 9.1.2. In order to achieve fast convergence the use of a shift is recommended, our implementation uses the shift-and-invert technique discussed in Section 8.3 with a complex shift. Using the shift $5.4 + 0.75i$, this approach yields the same spectral zeros and the same subspace as the Cayley approach, if a 10-dimensional subspace is sought. While `'eigs'` needed 22 iterations for convergence, the implicitly restarted symplectic Lanczos method converges after 1 iteration. Using the obtained subspace to compute a reduced order system, we obtain a reduced system which matches the original system quite well, see the Bode plots in Figure 9.4 and Figure 9.5.

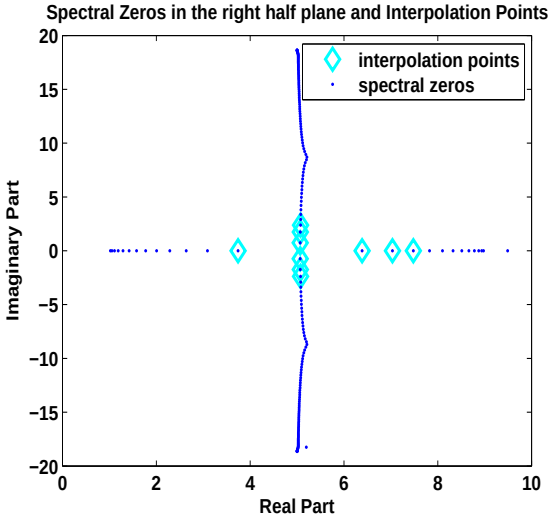


FIG. 9.3. *Spectral zeros and interpolation points*

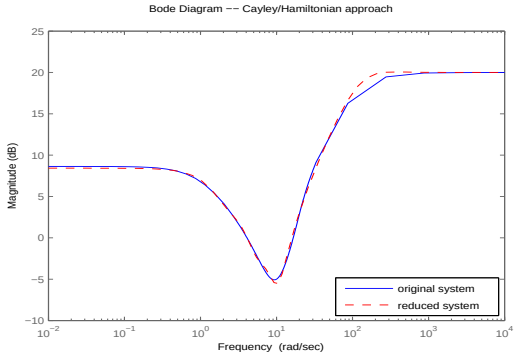


FIG. 9.4. *Bode plot comparing the original system and the reduced-order systems*

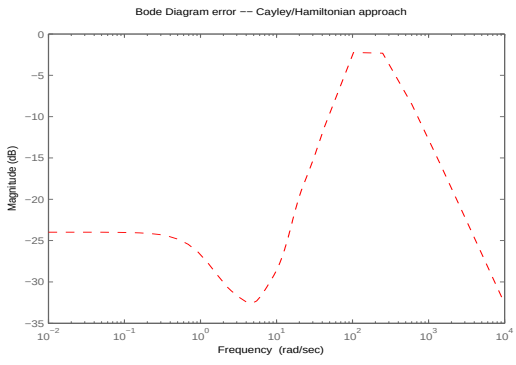


FIG. 9.5. *Bode plot for the error systems*

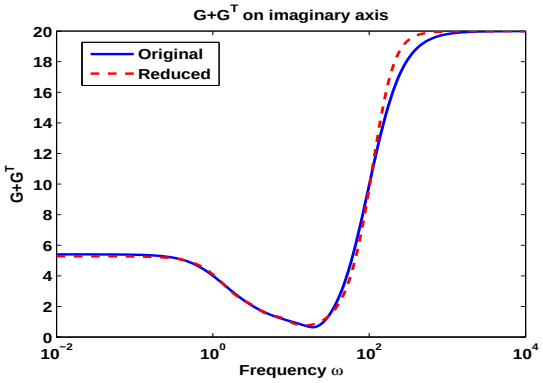


FIG. 9.6. *$G(i\omega) + G(-i\omega)$ on the imaginary axis*

The main computational effort for both approaches consists in the set up of the shifted operator and its application to a vector. For the implicitly restarted symplectic Lanczos method, the effort for setting up H_4

$$H_4 = H(H-\sigma I)^{-1}(H+\sigma I)^{-1}(H-\overline{\sigma} I)^{-1}(H+\overline{\sigma} I)^{-1} = (H^3-(\overline{\sigma}^2+\sigma^2)H+|\sigma|^4 H^{-1})^{-1}$$

and its application to a vector has been considered in Section 8.3: Once the LU decompositions $A-\tau I$ and $A+\tau I$ are available, the operator H_4 can be set up. This requires

- 16 triangular solves and
- 4 scalar products.

The application of the operator H_4 to a vector now requires

- 16 triangular solves,
- 1 matrix-vector product with A and 1 matrix-vector product with A^T,
- 16 scalar products and
- 16 saxpy operations.

For the implicitly restarted symplectic Lanczos method, the effort for setting up the Cayley approach operator

$$(\mu\mathcal{E} - \mathcal{A})^{-1}(\mu\mathcal{E} + \mathcal{A})$$

and its application to a vector is as follows: Once the LU decompositions $A - \tau I$ and $A + \tau I$ are available, the operator can be set up. This requires no additional operations. The application of the operator to a vector requires

- 8 triangular solves,
- 4 scalar products and
- 4 saxpy operations.

As in each of the 22 iteration steps of `'eigs'`, the operator has to be applied to a vector, this is clearly much more expensive then just 1 iteration step of the implicitly restarted symplectic Lanczos process, even if the overhead for setting up H_4 is taken into account.

The reduced system is stable and passive. The passivity can be observed in Figure 9.6, as the evaluation of $G(i\omega)+G(-i\omega)$ on the imaginary axis is always positive here.

While the Cayley approach allows only for real shift, the approach via the Hamiltonian eigenproblem allows to use complex shifts. For the shift $5+6i$, the interpolation points were chosen as depicted in Figure 9.7.

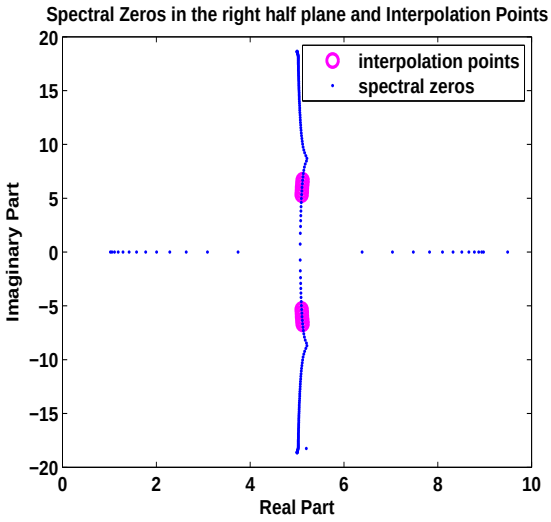


FIG. 9.7. *Spectral zeros and interpolation points*

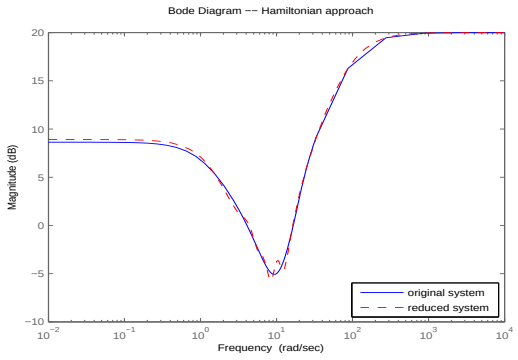


FIG. 9.8. *Bode plot comparing the original system and the reduced-order systems*

This results in a reduced system which approximates the original system much better than the one before, see the Bode plots in Figure 9.8 and Figure 9.9.

As before, the reduced system is stable and passive. The passivity can be observed in Figure 9.10, as the evaluation of $G(i\omega) + G(-i\omega)$ on the imaginary axis is always positive here.

These results are very limited. The motivation of this section was to demonstrate the possible application of the implicitly restarted symplectic Lanczos method in the context of passivity preserving model reduction. There are still a number of open problems for the passivity preserving model reduction problem which have not been addressed here. It is not clear how to choose the interpolation points and whether it is possible to derive a bound on the error of the reduced model.

9.2. Large-scale quadratic eigenvalue problems wth Hamiltonian eigenstructure. This section is concerned with the solution of the quadratic eigenvalue

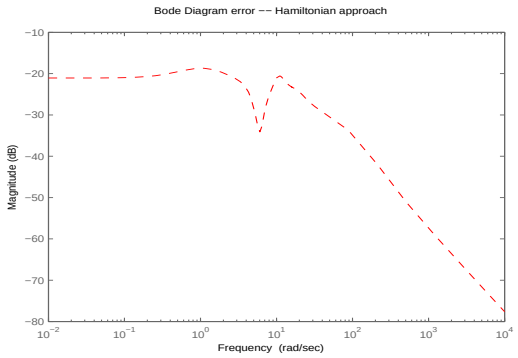


FIG. 9.9. *Bode plot for the error systems*

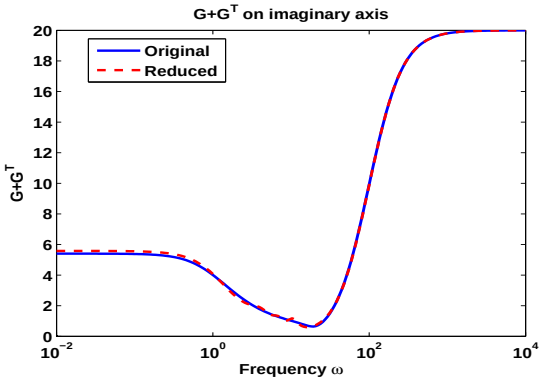


FIG. 9.10. *Pos Real*

problem (QEP)

$$(\lambda^2 M + \lambda G + K)x = 0, \quad M = M^T,\ G = -G^T,\ K = K^T, \tag{9.15}$$

where $M, G, K \in \mathbb{R}^{n\times n}$ are large and sparse. The task is to compute $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n \setminus \{0\}$ such that (9.15) holds. Such eigenvalue problems arise, for example, from finite element discretizations for analyzing the elastic deformation of anisotropic materials or computing corner singularities, see, e.g., [7, 113] and the references therein. In these applications M and $-K$ are positive definite mass and stiffness matrices, respectively. Gyroscopic systems are another source of quadratic eigenproblems (9.15). Here, M and K are positive definite mass and stiffness matrices and G is the gyroscopic matrix resulting from the Coriolis force. Such systems arise when modeling vibrations of spinning structures such as the simulation of tire noise, helicopter rotor blades, or spin-stabilized satellites with appended solar panels or antennas; see [54, 79, 135] and references therein. In the simulation of vibro-acoustics in flexible piping systems the coupling of the linear wave equation without internal flow and the structural Lamé-Navier equations at fluid-structure interfaces also leads to a quadratic eigenvalue problem with Hamiltonian symmetry [96]. A summary of conditions under which quadratic operator eigenvalue problems have a spectrum with Hamiltonian symmetry is given in [114].

Depending on the application, different parts of the spectrum are of interest. Typically, one is interested in the eigenvalues with smallest real part or the eigenvalues smallest or largest in modulus. The usual approach is to first linearize the quadratic eigenproblem to a generalized eigenproblem, transform this into a standard eigenproblem and solve that one using a (shift-and-invert) Krylov subspace method. There are also different approaches such as the SOAR (second-order Arnoldi) algorithm [9], a Krylov subspace based method for the solution of the quadratic eigenvalue problem, or the Jacobi-Davidson algorithm applied to polynomial eigenvalue problems [125]. Both approaches can be applied to the quadratic eigenproblem directly without any linearization.

Here we will be concerned with algorithms for solving the quadratic eigenvalue problem which preserve and exploit its Hamiltonian eigenstructure. For this, we will make use of the linearization discussed in Section 8.3 which leads to the Hamiltonian matrix H (8.19)

$$H = \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix}.$$

Typical applications require a few eigenvalues that are largest or smallest in magnitude or closest to the imaginary axis. Computing the ones largest in magnitude, can be achieved efficiently by Krylov subspace methods, e.g., Arnoldi or Lanczos processes, possibly combined with implicit restarting or a Krylov-Schur-type technique [131]. To compute other eigenvalues, first transformations must be applied to the matrix H which have the effect of shifting the desired eigenvalues to the periphery of the spectrum. In light of the symmetry of the spectrum, one might think of working with $(H - \tau I)^{-1}(H + \tau I)^{-1}$, in case τ is real or purely imaginary. All eigenvalues near to $\pm\tau$ are mapped simultaneously to values of large modulus. But this matrix is not Hamiltonian but skew-Hamiltonian. The standard (implicitly restarted) Arnoldi method automatically preserves this structure. This led to the development of the SHIRA method as a structure-preserving (shift-and-invert) Arnoldi method for Hamiltonian matrices [101].

Here we discuss an alternative, the restarted symplectic Lanczos algorithm, the Hamiltonian Krylov-Schur-type method as discussed in Section 8.5. In order to stay within the Hamiltonian structure, for a real shift τ, we can work with the Hamiltonian matrix

$$H_1(\tau) = H^{-1}(H - \tau I)^{-1}(H + \tau I)^{-1} = (H^3 - \tau^2 H)^{-1}, \tag{9.16}$$

or

$$H_2(\tau) = H(H - \tau I)^{-1}(H + \tau I)^{-1} = (H - \tau^2 H^{-1})^{-1}, \tag{9.17}$$

for example. In case a complex shift τ is used, we can work with the Hamiltonian matrix

$$\begin{aligned} H_3(\tau) &= H^{-1}(H - \tau I)^{-1}(H + \tau I)^{-1}(H - \overline{\tau} I)^{-1}(H + \overline{\tau} I)^{-1} \\ &= (H^5 - (\overline{\tau}^2 + \tau^2)H^3 + |\tau|^4 H)^{-1} \end{aligned} \tag{9.18}$$

or

$$\begin{aligned} H_4(\tau) &= H(H - \tau I)^{-1}(H + \tau I)^{-1}(H - \overline{\tau} I)^{-1}(H + \overline{\tau} I)^{-1} \\ &= (H^3 - (\overline{\tau}^2 + \tau^2)H + |\tau|^4 H^{-1})^{-1}. \end{aligned} \tag{9.19}$$

The shift-and-invert operators $H_1(\tau), H_2(\tau), H_3(\tau)$ have first been considered in [113, 139] while $H_4(\tau)$ is examined in [20].

When largest or smallest modulus eigenvalues are required, the Hamiltonian Krylov-Schur-type algorithm is applied to H as in (8.19) or

$$H^{-1} = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -K^{-1} \\ M & 0 \end{bmatrix} \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix}. \tag{9.20}$$

As SHIRA requires a skew-Hamiltonian operator, H^2 or H^{-2} have to be used in these situations so that the Hamiltonian Krylov-Schur-type approach is more efficient in these cases.

When other eigenvalues are required, H_2 or H_4 will be used. In this situation SHIRA will work with a similar, so skew-Hamiltonian operator. The cost for applying the operators is the same, so that in this respect, none of the algorithms is to be preferred.

An advantage of the Hamiltonian Krylov-Schur-type method is that the lower half of the computed eigenvectors of H and H^{-1} yields the corresponding eigenvector of the quadratic matrix polynomial $Q(\lambda)$ while the eigenvectors computed by SHIRA can not be used to deduce eigenvectors of $Q(\lambda)$; see [7, 101, 113] for more details on this. In any case, no matter how approximate eigenvalues $\tilde{\lambda}$ are computed, eigenvectors can also be obtained by applying the inverse iteration

$$Q(\tilde{\lambda})x_k = x_{k-1}, \quad x_k = \frac{1}{\|x_k\|}x_k, \quad k = 1, 2, \ldots, \tag{9.21}$$

where x_0 of unit norm is chosen arbitrarily, see [113, 132].

9.2.1. Numerical results. In this section, we report the results of numerical experiments obtained with the Krylov-Schur-type method for Hamiltonian eigenproblems applied to the QEP (8.18). All experiments are performed in MATLAB R2006a using double precision on a Pentium M notebook with 512 MB main memory or a HP compute server with 2 Xeon 3,06 GHz processors, 533 MHz 512-KB level 2 cache, 1 MB level 3 cache, 9 GB main memory (of which only 2 GB can be used due to MATLAB's limitations in addressing more than 2 GB of memory in the available 32-Bit version of MATLAB).

The accuracy of computed eigenvalues and eigenvectors is compared using relative residuals

$$\frac{\|Q(\tilde{\lambda})\tilde{x}\|_1}{\|Q(\tilde{\lambda})\|_1\|\tilde{x}\|_1},$$

where $(\tilde{\lambda}, \tilde{x})$ is a computed Ritz pair.

Computing corner singularities. The solutions of elliptic boundary value problems like the Laplace and linear elasticity (Lamé) equations in domains with polyhedral corners often exhibit singularities in the neighborhood of the corners. The singularities can be quantified if this neighborhood is intersected with the unit ball centered at the corner and parameterized with spherical coordinates (r, ϕ, θ). Then the singular part of the solution can be expanded in a series with terms of the form $r^\alpha u(\phi, \theta)$, where α is the singular exponent. It turns out that $\alpha = \lambda - \frac{1}{2}$ and u can be computed as eigenpairs of quadratic operator eigenvalue problems of the form

$$\lambda^2 m(u, v) + \lambda g(u, v) = k(u, v), \tag{9.22}$$

SHIRA 3 iterations	
Eigenvalue	Residual
-0.90592878886122	$6.945 \cdot 10^{-17}$
-0.90634686034999	$1.325 \cdot 10^{-16}$
-1.07560224930036	$3.933 \cdot 10^{-17}$
-1.60332758477172	$1.639 \cdot 10^{-15}$
-1.65786577098970	$6.311 \cdot 10^{-16}$
-1.66121735256372	$2.157 \cdot 10^{-16}$

TABLE 9.1
Fichera corner ($n = 5139$): SHIRA with shift $\tau = 1$, 12 eigenvalues requested.

where $m(.,.), k(.,.)$ are Hermitian positive definite sesquilinear forms and $g(.,.)$ is a skew-Hermitian sesquilinear form. Finite-element discretization of the operator eigenvalue problem (9.22) leads to a QEP as in (8.18), where M and $-K$ are positive definite. For the numerical solution of (9.22), the software package *CoCoS*[4] [113] can be used. Note that *CoCoS* includes Fortran implementations of SHIRA as well as a solver based on applying the implicitly restarted Hamiltonian Lanczos process [17, 139] to H^{-1} from (9.20). Here we use *CoCoS* only for the assembly of the matrices M, G, K.

In the following example we consider brittle elastic bodies where the environment of crack peaks is sufficiently well approximated by the linear material law (Lamé equation). A more detailed discussion can be found in [116]. The 3D elasticity problem is considered for the Fichera corner which results from cutting the cube $[0,1] \times [0,1] \times [0,1]$ out of the cube $(-1,1) \times (-1,1) \times (-1,1)$. The problem is defined by the Lamé constants μ, ν derived from Young's modulus of elasticity, and the opening angle ξ of the corner. In the following computations for the Fichera corner, Lamé's constants are $\mu = 0.5$ and $\nu = 0.3$ and the opening angle is $90°$. We compare the Hamiltonian Krylov-Schur-type algorithm, a MATLAB implementation of SHIRA and the MATLAB function `eigs`.

For the first test an example of size $n = 5139$ is chosen. All algorithms use the same starting vector. We chose the shift $\tau = 1$ and set up an operator to apply $H_2(\tau)$ to a vector. In order to compare the three algorithms considered here, the Hamiltonian Krylov-Schur-type algorithm and MATLAB's `eigs` are applied to $H_2(1)$, while SHIRA is used with the skew-Hamiltonian operator $R_2(1)$ as in (8.20). We asked for 12 eigenvalues and allowed a search space of size 24. The results are shown in Tables 9.1 and 9.2. Here and in the following we only show results for the negative eigenvalues; results for their positive counterparts are similar. In both cases the residuals are computed using eigenvectors obtained from inverse iteration as in (9.21). The results are comparable: SHIRA needs the least iterations to meet its stopping criterion, but some of the eigenvalues are slightly less accurate than for the other two methods. On the other hand, the Hamiltonian Krylov-Schur-type method needs two iterations less than `eigs` at similar accuracy.

Next we compare the Hamiltonian Krylov-Schur-type method and `eigs` for computing some eigenvalues of smallest magnitude; hence we apply them to H^{-1}. As before, we are interested in 12 eigenvalues and the search space has the size 24. Both algorithms get the same starting vector. The results are shown in the Table 9.3.

[4] See `http://www-user.tu-chemnitz.de/~co/cocos/project/cocos.php`

`eigs` 6 iterations		HKS 4 iterations max. condition number 26537	
Eigenvalue	Residual	Eigenvalue	Residual
-0.90592878886137	$4.2 \cdot 10^{-17}$	-0.90592878886208	$6.7 \cdot 10^{-17}$
-0.90634686034987	$4.8 \cdot 10^{-17}$	-0.90634686034995	$4.7 \cdot 10^{-17}$
-1.07560224930041	$4.7 \cdot 10^{-17}$	-1.07560224930041	$4.6 \cdot 10^{-17}$
-1.60332758476362	$2.5 \cdot 10^{-17}$	-1.60332758476305	$1.3 \cdot 10^{-16}$
-1.65786577098499	$2.8 \cdot 10^{-17}$	-1.65786577098420	$1.0 \cdot 10^{-16}$
-1.66121735256606	$2.6 \cdot 10^{-17}$	-1.66121735256562	$4.9 \cdot 10^{-17}$

TABLE 9.2

Fichera corner ($n = 5139$): `eigs` and Hamiltonian Krylov-Schur (HKS) applied to $H_2(1)$, 12 eigenvalues requested.

`eigs` 8 iterations		HKS 6 iterations max. condition number 4298		
Eigenvalue	Residual	Eigenvalue	Residual	
-0.90592878886009	$8.9 \cdot 10^{-17}$	-0.90592878886360	$1.4 \cdot 10^{-16}$	$2.9 \cdot 10^{-15}$
-0.90634686034887	$9.3 \cdot 10^{-17}$	-0.90634686035032	$5.5 \cdot 10^{-17}$	$2.6 \cdot 10^{-15}$
-1.07560224929966	$5.8 \cdot 10^{-17}$	-1.07560224929977	$5.2 \cdot 10^{-17}$	$2.9 \cdot 10^{-15}$
-1.60332758476358	$2.8 \cdot 10^{-17}$	-1.60332758476361	$2.6 \cdot 10^{-17}$	$3.6 \cdot 10^{-15}$
-1.65786577098385	$1.4 \cdot 10^{-16}$	-1.65786577098420	$1.0 \cdot 10^{-16}$	$3.3 \cdot 10^{-14}$
-1.66121735256510	$1.1 \cdot 10^{-16}$	-1.66121735256192	$4.8 \cdot 10^{-16}$	$5.3 \cdot 10^{-14}$

TABLE 9.3

Fichera corner ($n = 5139$): `eigs` and Hamiltonian Krylov-Schur (HKS) applied to H^{-1}, 12 eigenvalues requested. The left residuals in the HKS column are computed using eigenvectors obtained by inverse iteration, the right residuals correspond to eigenvectors read off of the lower half of the Ritz vectors of H^{-1}.

Comparing the results it is easy to see that the number of iterations and the residuals are very similar. In general, the Hamiltonian Krylov-Schur-type method needs fewer iterations than `eigs` while yielding the same accuracy. Here we also compare residuals obtained for eigenvectors computed by the inverse iteration (9.21) and read off of the Ritz vectors of H^{-1} computed within the symplectic Lanczos process. The latter ones are clearly less accurate, but on the other hand they are obtained as by-products and do not require any factorization of $Q(\tilde{\lambda})$, which is the most expensive part in all of the computations!

Next we choose an example in which the matrices M, G and K are all of size $n = 12828$. The same kind of test runs as above are reported in the Tables 9.4, 9.5 and 9.6.

Gyroscopic systems. Such systems arise when modeling vibrations of spinning structures such as the simulation of tire noise, helicopter rotor blades, or spin-stabilized satellites with appended solar panels or antennas; see [54, 79, 135] and references therein. Here, M and K are positive definite mass and stiffness matrices and G is the gyroscopic matrix resulting from the Coriolis force. It is known that under these conditions, all eigenvalues of the QEP are purely imaginary, see [80, 135].

For our experiments, we chose the model of a butterfly gyro as described in [30].

SHIRA 3 iterations	
Eigenvalues	Residuals
-0.90510929898159	$1.810 \cdot 10^{-16}$
-0.90529568786501	$2.778 \cdot 10^{-16}$
-1.07480595544986	$6.573 \cdot 10^{-17}$
-1.60117345104856	$1.414 \cdot 10^{-15}$
-1.65765608689995	$4.930 \cdot 10^{-15}$
-1.65914529725448	$1.994 \cdot 10^{-15}$

TABLE 9.4

Fichera corner ($n = 12828$): SHIRA with shift $\tau = 1$, 12 eigenvalues requested.

`eigs` 6 iterations		HKS 4 iterations max. condition number 335526	
Eigenvalue	Residual	Eigenvalue	Residual
-0.90510929898127	$8.2 \cdot 10^{-17}$	-0.90510929894951	$4.7 \cdot 10^{-16}$
-0.90529568786417	$7.4 \cdot 10^{-17}$	-0.90529568784944	$2.7 \cdot 10^{-16}$
-1.07480595545002	$7.2 \cdot 10^{-17}$	-1.07480595544985	$7.0 \cdot 10^{-17}$
-1.60117345102312	$8.7 \cdot 10^{-17}$	-1.60117345101134	$5.8 \cdot 10^{-16}$
-1.65765608688689	$7.3 \cdot 10^{-17}$	-1.65765608679830	$2.6 \cdot 10^{-15}$
-1.65914529726339	$3.7 \cdot 10^{-16}$	-1.65914529702482	$7.4 \cdot 10^{-15}$

TABLE 9.5

Fichera corner ($n = 12828$): `eigs` and Hamiltonian Krylov-Schur (HKS) applied to $H_2(1)$, 12 eigenvalues requested

`eigs` 8 iterations		HKS 6 iterations max. condition number 4298		
Eigenvalue	Residual	Eigenvalue	Residual	
-0.90510929897842	$1.1 \cdot 10^{-16}$	-0.90510929890243	$1.1 \cdot 10^{-15}$	$7.2 \cdot 10^{-15}$
-0.90529568785959	$1.3 \cdot 10^{-16}$	-0.90529568782464	$6.2 \cdot 10^{-16}$	$6.1 \cdot 10^{-15}$
-1.07480595544558	$1.2 \cdot 10^{-16}$	-1.07480595544473	$1.4 \cdot 10^{-16}$	$7.6 \cdot 10^{-16}$
-1.60117345102137	$4.8 \cdot 10^{-17}$	-1.60117345100988	$6.5 \cdot 10^{-16}$	$1.1 \cdot 10^{-12}$
-1.65765608688686	$7.1 \cdot 10^{-17}$	-1.65765608828382	$4.1 \cdot 10^{-14}$	$1.9 \cdot 10^{-10}$
-1.65914529726390	$5.2 \cdot 10^{-17}$	-1.65914530264081	$1.7 \cdot 10^{-13}$	$2.9 \cdot 10^{-10}$

TABLE 9.6

Fichera corner ($n = 12828$): `eigs` and Hamiltonian Krylov-Schur (HKS) applied to H^{-1}, 12 eigenvalues requested. The left residuals in the HKS column are computed using eigenvectors obtained by inverse iteration, the right residuals correspond to eigenvectors read off of the lower half of the Ritz vectors of H^{-1}.

The butterfly gyro is a vibrating micro-mechanical system developed for use in inertial navigation applications.

The data matrices M, K of order $n = 17361$ (which are available from the Oberwolfach Benchmark Collection[5]) are obtained from a finite-element analysis performed

[5]See `http://www.imtek.de/simulation/benchmark`.

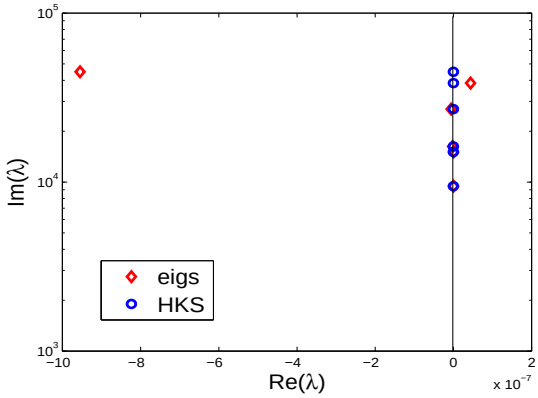


FIG. 9.11. *Butterfly gyro ($n = 17361$):* `eigs` *and Hamiltonian Krylov-Schur (HKS) applied to* H^{-1}*, 12 eigenvalues requested.*

with ANSYS using quadratic tetrahedral elements (SOLID187). As the gyroscopic matrix G is missing, we choose a randomly generated skew-symmetric matrix with the same sparsity pattern as K and with entries of considerably smaller magnitude as the influence of the Coriolis force is usually much smaller than that of the stiffness of the system.

In our first test run, we apply `eigs` and the Hamiltonian Krylov-Schur-type method to H^{-1} to obtain the smallest frequency modes of the butterfly gyro—these are usually the modes of interest in vibration analysis of gyroscopic systems. Both the Hamiltonian Krylov-Schur-type method and `eigs` need 3 iterations to compute 12 eigenvalues. The maximal condition number encountered in the SR algorithm is about $1.5 \cdot 10^3$. The accuracy of the computed eigenvalues as measured by their relative residuals is similar: for both methods, the residuals are smaller than the machine epsilon `eps`. Figure 9.11 shows the computed eigenvalues with positive imaginary parts. Obviously, the Hamiltonian Krylov-Schur-type method locates all eigenvalues on the imaginary axis as expected from theory while for the eigenvalues computed by `eigs`, it is rather difficult to decide whether or not these are purely imaginary even so the real parts are relatively small.

In the second test run for the butterfly gyro, we try to compute interior eigenvalues. For this purpose, we choose a shift $\tau = 10^6 i$ and applied `eigs` and the Hamiltonian Krylov-Schur-type method to $H_2(\tau)$. Here, `eigs` needs one iteration less than the Krylov-Schur-type method (2 as compared to 3). The maximum condition number encountered in the SR algorithm is about $3.15 \cdot 10^6$. The accuracy of the computed eigenvalues seems to be not affected by this fairly large condition number: again, all residuals are smaller than 10^{-16} as for `eigs`. But again, the new approach yields physically meaningful results as all computed eigenvalues are located on the imaginary axis. In contrast, the eigenvalues computed by `eigs` have nonzero real parts as can be seen from Figure 9.12 except for the eigenvalue closest to the target $\tau = 10^6 i$.

9.3. Conclusions. We have discussed the application of the Hamiltonian Krylov-Schur-type method based on the symplectic Lanczos process to quadratic eigenvalue problems with Hamiltonian symmetry. The method is an alternative to unstructured methods like the implicitly restarted Arnoldi method as implemented in the MAT-

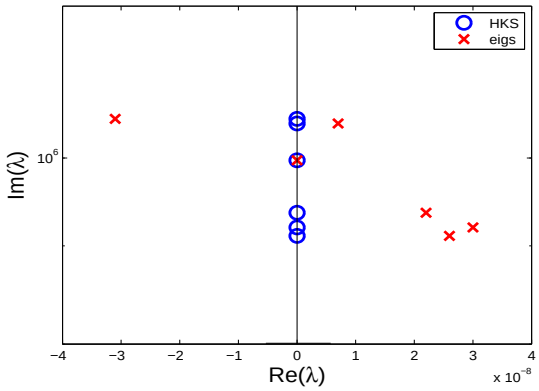


FIG. 9.12. *Butterfly gyro ($n = 17361$):* `eigs` *and Hamiltonian Krylov-Schur (HKS) applied to $H_2(10^6 i)$, 12 eigenvalues requested.*

LAB function `eigs` or its structure-preserving variant SHIRA which can be applied to skew-Hamiltonian operators. Compared to `eigs` our method has the advantage of respecting the symmetry properties inherent to the problem and thus yields meaningful physical results. This is demonstrated for a stable gyroscopic system where theoretically all eigenvalues are located on the imaginary axis. On the other hand, in several situations the Hamiltonian Krylov-Schur-type approach is more efficient than SHIRA. In particular, eigenvectors are directly available in case no shifts are used while SHIRA only provides eigenvalues.

Acknowledgments My thanks go to Peter Benner, Nil Mackey and Michael Overton for their hospitality in Chemnitz, Kalamazoo and New York during my sabbatical. I would like to point out that the implementation of the SR algorithm in [132] is nearly parameterized, although the implementation sets up and uses the Hamiltonian J-Hessenberg matrix as an $2n \times 2n$ matrix. We thank Conny Pester for providing the data matrices for the Fichera corner examples.

REFERENCES

[1] G.S. 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.

[2] G. AMMAR, C. MEHL, AND V. MEHRMANN, *Schur-like forms for matrix Lie groups, Lie algebras and Jordan algebras*, Linear Algebra Appl., 287 (1999), pp. 11–39.

[3] G.S. 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 Publications, Philadelphia, PA, 2nd ed., 1995.

[5] A.C. ANTOULAS, *A new result on positive real interpolation and model reduction*, Systems & Control Letters, 54 (2005), pp. 361–374.

[6] ———, *A new result on positive real interpolation and model reduction*, Systems & Control Letters, 54 (2005), pp. 361–374.

[7] T. APEL, V. MEHRMANN, AND D.S. WATKINS, *Structured eigenvalue methods for the computation of corner singularities in 3d anisotropic elastic structures*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4459–4473.

[8] Z. BAI, *Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem*, Mathematics of Computation, 62 (1994), pp. 209–226.

[9] Z. Bai and Y. Su, *Soar: A second-oder Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal., 26 (2005), pp. 640–659.
[10] N.E. Barabanov, A.Kh. Gelig, G.A. Leonov, A.L. Likhtarnikov, A.S. Matveev, V.B. Smirnova, and A.L. Fradkov, *The frequency theorem (kalman-yakubovich lemma) in control theory*, Automat. Remote Control, 57 (1996), pp. 1377–1407.
[11] C. Beattie, M. Embree, and J. Rossi, *Convergence of restarted Krylov subspaces to invariant subspaces*, SIAM J. Matrix Anal. Appl., 35 (2004), pp. 1074–1109.
[12] P. Benner, *Computational methods for linear-quadratic optimization*, Supplementario ai Rendiconti del Circolo Matematico di Palermo, Serie II (1997), pp. 21–56.
[13] ———, *Symplectic balancing of Hamiltonian matrices*, SIAM J. Sci. Statist. Comput., 22 (2000), pp. 1885–1904.
[14] P. Benner, R. Byers, V. Mehrmann, and H. Xu, *Numerical methods for linear quadratic and H_∞ control problems*, in Dynamical systems, control, coding, computer vision: new trands, interfaces, and interplay, G. Picci and D.S. Gilliam, eds., vol. 25 of Progress in systems and control theory, Birkhäuser, Basel, 1999, pp. 203–222.
[15] ———, *Robust numerical methods for robust control*, Technical Report 06–2004, Institut für Mathematik, TU Berlin, Germany, 2004.
[16] P. Benner and H. Fassbender, *A restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Technical Report SPC 95_28, TU Chemnitz-Zwickau (now TU Chemnitz), Fakultät für Mathematik, Germany, 1995.
[17] ———, *An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Linear Algebra Appl., 263 (1997), pp. 75–111.
[18] ———, *An implicitly restarted Lanczos method for the symplectic eigenvalue problem*, Berichte aus der Technomathematik, Report 98–01, Zentrum für Technomathematik, FB 3 – Mathematik und Informatik, Universität Bremen, 28334 Bremen, FRG, 1998.
[19] ———, *Computing passive reduced-order LTI models using structured Krylov subspace methods*, preprint, TU Braunschweig, Germany, 2006.
[20] P. Benner, H. Fassbender, and M. Stoll, *A Hamiltonian Krylov-Schur-type method based on the symplectic Lanczos process*, in preparation.
[21] P. Benner, H. Fassbender, and D.S. Watkins, *Two connections between the SR and HR eigenvalue algorithms*, Linear Algebra Appl., 272 (1997), pp. 17–32.
[22] P. Benner and D. Kressner, *Balancing sparse Hamiltonian matrices*, Linear Algebra Appl., 415 (2006), pp. 3–19.
[23] ———, *Algorithm 8xx: Fortran 77 subroutines for computing the eigenvalues of Hamiltonian matrices*, ACM Transactions on Mathematical Software, (to appear).
[24] P. Benner, A. Laub, and V. Mehrmann, *A collection of benchmark examples for the numerical solution of algebraic Riccati equations II: Discrete-time case*, Tech. Report SPC 95_23, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1995. Available from `http://www.tu-chemnitz.de/sfb393/spc95pr.html`.
[25] P. Benner, V. Mehrmann, and D.C. Sorensen, *Dimension Reduction of Large-Scale Systems*, vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, 2005.
[26] P. Benner, V. Mehrmann, and H. Xu, *A new method for computing the stable invariant subspace of a real Hamiltonian matrix*, J. Comput. Appl. Math., 86 (1997), pp. 17–43.
[27] ———, *A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils*, Numerische Mathematik, 78 (1998), pp. 329–358.
[28] ———, *A note on the numerical solution of complex Hamiltonian and skew-Hamiltonian eigenvalue problems*, Electr. Trans. Num. Anal., 8 (1999), pp. 115–126.
[29] R. Bhatia, *Matrix factorizations and their perturbations*, Linear Algebra Appl., 197–198 (1994), pp. 245–276.
[30] D. Billger, *The butterfly gyro*, in Dimension Reduction of Large-Scale Systems, P. Benner, V. Mehrmann, and D.C. Sorensen, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin/Heidelberg, Germany, 2005, pp. 349–352.
[31] S. Boyd, V. Balakrishnan, and P. Kabamba, *A bisection method for computing the H_∞ norm of a transfer matrix and related problems*, Math. Control, Signals, Sys., 2 (1989), pp. 207–219.
[32] M.A. Brebner and J. Grad, *Eigenvalues of $Ax = \lambda Bx$ for real symmetric matrices A and B computed by reduction to pseudosymmetric form and the HR process*, Linear Algebra Appl., 43 (1982), pp. 99–118.
[33] W. Bunse and A. Bunse-Gerstner, *Numerische lineare Algebra*, Teubner, 1985.
[34] A. Bunse-Gerstner, *Berechnung der Eigenwerte einer Matrix mit dem HR-Verfahren*, in Numerische Behandlung von Eigenwertaufgaben, Band 2, Birkhäuser Verlag Basel, 1979,

pp. 26–39.
[35] A. Bunse-Gerstner, *An analysis of the HR algorithm for computing the eigenvalues of a matrix*, Linear Algebra Appl., 35 (1981), pp. 155–173.
[36] A. Bunse-Gerstner, *Matrix factorizations for symplectic QR-like methods*, Linear Algebra Appl., 83 (1986), pp. 49–77.
[37] A. Bunse-Gerstner and H. Fassbender, *A Jacobi-like method for solving algebraic Riccati equations on parallel computers*, IEEE Trans. Automat. Control, 42 (1997), pp. 1071–1087.
[38] 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.
[39] A. Bunse-Gerstner, V. Mehrmann, and D.S. Watkins, *An SR algorithm for Hamiltonian matrices based on Gaussian elimination*, Methods of Operations Research, 58 (1989), pp. 339–356.
[40] J.V. Burke, A.S. Lewis, and M.L. Overton, *Robust stability and a criss-cross algorithm for pseudospectra*, IMA J. Numer. Anal., 23 (2003), pp. 359–375.
[41] R. Byers, *Hamiltonian and symplectic algorithms for the algebraic Riccati equation*, PhDthesis, Cornell University, Dept. Comp. Sci., Ithaca, NY, 1983.
[42] ———, *A Hamiltonian QR-algorithm*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 212–229.
[43] ———, *A bisection method for measuring the distance of a stable to unstable matrices*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 875–881.
[44] ———, *A Hamiltonian-Jacobi algorithm*, IEEE Trans. Automat. Control, 35 (1990), pp. 566–570.
[45] R. Byers and D. Kressner, *Structured condition numbers for invariant subspaces*, SIAM J. Matrix Anal. Appl., (to appear).
[46] D. Calvetti, L. Reichel, and D.C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electr. Trans. Num. Anal., 2 (1994), pp. 1–21.
[47] X.-W. Chang, *On the sensitivity of the SR decomposition*, Linear Algebra Appl., 282 (1998), pp. 297–310.
[48] D. Chu, X. Liu, and V. Mehrmann, *A numerically strongly stable method for computing the Hamiltonian Schur form*, to appear in Numer. Math., (2006).
[49] T.A. Davis, *Umfpack version 4.4 user guide*, tech. report, Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, 2005.
[50] J. Della-Dora, *Sur quelques Algorithmes de recherche de valeurs propres*, Thése, L'Université Scientifique et Medicale de Grenoble, 1973.
[51] ———, *Numerical linear algorithms and group theory*, Linear Algebra Appl., 10 (1975), pp. 267–283.
[52] J.W. Demmel, J.r. Gilbert, and X.S. Li, *Superlu users' guide*, tech. report, Lawrence Berkeley National Laboratory, 2003.
[53] L. Elsner, *On some algebraic problems in connection with general eigenvalue algorithms*, Linear Algebra Appl., 26 (1979), pp. 123–138.
[54] K. Elssel and H. Voss, *Reducing huge gyroscopic eigenproblems by automated multi-level substructuring*, Arch. Appl. Mech., 76 (2006), pp. 171–179.
[55] H. Fassbender, *Symplectic methods for the symplectic eigenproblem*, Kluwer Academic/Plenum Publishers, New York, 2000.
[56] H. Fassbender, D.S. Mackey, and N. Mackey, *Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems*, Linear Algebra Appl., 332-334 (2001), pp. 37–80.
[57] W.R. Ferng, W.-W. Lin, and C.-S. Wang, *The shift-inverted J-Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations*, Comput. Math. Appl., 33 (1997), pp. 23–40.
[58] ———, *Errata to 'the shift-inverted J-Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations'*, Comput. Math. Appl., 38 (1999), pp. 253–255.
[59] J.G.F. Francis, *The QR transformation, Part I and Part II*, Comput. J., 4 (1961), pp. 265–271 and 332–345.
[60] G. Freiling, V. Mehrmann, and H. Xu, *Existence, uniqueness and parameterization of Lagrangian invariant subspaces*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 1045–1069.
[61] R.W. Freund, *Transpose-free quasi-minimal residual methods for non-Hermitian linear systems*, in Recent advances in iterative methods. Papers from the IMA workshop on iterative methods for sparse and structured problems, held in Minneapolis, MN, February 24-March 1, 1992., G. Golub et al., ed., vol. 60 of IMA Vol. Math. Appl., New York, NY, 1994, Springer–Verlag, pp. 69–94.

[62] ———, *Model reduction methods based on krylov subspaces*, Acta Numerica, 12 (2003), pp. 267–319.
[63] R.W. Freund, M.H. Gutknecht, and N.M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.
[64] R. Freund and V. Mehrmann, *A symplectic look-ahead Lanczos algorithm for the Hamiltonian eigenvalue problem*. manuscript.
[65] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 3rd ed., 1996.
[66] E.J. Grimme, D.C. Sorensen, and P. Van Dooren, *Model reduction of state space systems via an implicitly restarted Lanczos method*, Numer. Algorithms., 12 (1996), pp. 1–31.
[67] S. Gugercin and A.C. Antoulas, *A survey of model reduction by balanced truncation and some new results*, International Journal of Control, 77 (2004), pp. 748–766.
[68] C-H. Guo and P. Lancaster, *Analysis and modification of Newton's method for algebraic riccati equations*, Math. Comp., 67 (1998), pp. 1089–1105.
[69] M. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part I*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.
[70] ———, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part II*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 15–58.
[71] A.S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964.
[72] Z. Jia, *The convergence of generalized Lanczos methods for large unsymmetric eigenproblems*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 843–862.
[73] W. Kahan, B.N. Parlett, and E. Jiang, *Residual bounds on approximate eigensystems of nonnormal matrices*, SIAM J. Numer. Anal., 19 (1982), pp. 470–484.
[74] M. Karow, D. Kressner, and F. Tisseur, *Structured eigenvalue condition numbers*, Numerical Analysis Report No. 467, Manchester Centre for Computational Mathematics, Manchester, England, (2005).
[75] M.M. Konstantinov, V. Mehrmann, and P.Hr. Petkov, *Perturbation analysis for the Hamiltonian Schur and block-Schur forms*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 387–424.
[76] D. Kressner, *Numerical Methods for General and Structured Eigenvalue Problems*, vol. 46 of Lecture Notes in Computational Science and Engineering, Springer, Berlin, 2005.
[77] V.N. Kublanoskaja, *On some algorithms for the solution of the complete eigenvalue problem*, U.S.S.R. Comput. Math. and Math. Phys., 3 (1961), pp. 637–657.
[78] A. Kugi and K. Schlacher, *Analyse und Synthese nichtlinearer dissipativer Systeme: Ein Überblick*, Automatisierungstechnik, 50 (2002), pp. 63–69 and 103–111.
[79] P. Lancaster, *Lambda-Matrices and Vibrating Systems*, Pergamon Press, Oxford, UK, 1966.
[80] ———, *Strongly stable gyroscopic systems*, Electr. J. Linear Algebra, 5 (1999), pp. 53–66.
[81] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995.
[82] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.
[83] A.J. 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).
[84] ———, *Invariant subspace methods for the numerical solution of Riccati equations*, in The Riccati Equation, S. Bittanti, A.J. Laub, and J.C. Willems, eds., Springer-Verlag, Berlin, 1991, pp. 163–196.
[85] A.J. Laub and K.R. Meyer, *Canonical forms for symplectic and Hamiltonian matrices*, Celestial Mechanics, 9 (1974), pp. 213–238.
[86] R.B. Lehoucq, *Analysis and Implementation of an implicitly restarted Arnoldi Iteration*, PhD thesis, Rice University, Dep. Computational and Applied Mathematics, Houston, Texas, 1995.
[87] ———, *On the convergence of an implicitly restarted Arnoldi method*, tech. report, Sandia National Laboratory, P.O. Box 5800, MS 1110, Albuquerque, NM 87185-1110, 1999.
[88] ———, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.
[89] R.B. Lehoucq and D.C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.
[90] R.B. Lehoucq, D.C. Sorensen, and C. Yang, *ARPACK user's guide. Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, SIAM Publications, Philadelphia,PA, 1998.

[91] W.-W. Lin and T.-C. Ho, *On Schur type decompositions for Hamiltonian and symplectic pencils*, tech. report, Institute of Applied Mathematics, National Tsing Hua University, 1990.
[92] W.-W. Lin, V. Mehrmann, and H. Xu, *Canonical forms for Hamiltonian and symplectic matrices and pencils*, Linear Algebra Appl., 302-303 (1999), pp. 469–533.
[93] Z.-S. Liu, *On the extended HR algorithm*, tech. report, Center for Pure and Applied Mathematics, University of California, Berkeley, 1992.
[94] L. Lopez and V. Simoncini, *Preserving geometric properties of the exponential matrix by block Krylov subspace methods*, Preprint, (2005).
[95] D.S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Vector spaces of linearizations for matrix polynomials*, SIAM J. Matrix Anal., (to appear).
[96] M. Maess and L. Gaul, *Simulation of vibro-acoustics in flexible piping systems*, GAMM Mitteilungen, 28 (2005), pp. 37–55.
[97] V. Mehrmann, *Der SR-Algorithmus zur Berechnung der Eigenwerte einer Matrix*, Diplomarbeit, Universität Bielefeld, Bielefeld, FRG, 1979.
[98] ———, *The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution*, no. 163 in Lecture Notes in Control and Information Sciences, Springer-Verlag, Heidelberg, 1991.
[99] V. Mehrmann and E. Tan, *Defect correction methods for the solution of algebraic Riccati equations*, IEEE Trans. Automat. Control, 33 (1988), pp. 695–698.
[100] V. Mehrmann and D.S. Watkins, *Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils*, SIAM J. Sci. Statist. Comput., 22 (2000), pp. 1905–1925.
[101] ———, *Structure-preserving methods for computing pairs of large sparse skew-Hamiltonian/Hamiltonian pencils*, SIAM J. Sci. Statist. Comput., 22 (2001), pp. 1905–1925.
[102] V. Mehrmann and H. Xu, *Canonical forms for Hamiltonian and symplectic matrices and pencils*, Tech. Report SFB393/98–07, Fak. f. Mathematik, TU Chemnitz–Zwickau, 09107 Chemnitz, FRG, 1998.
[103] Gao Mei, *A new method for solving the algebraic Riccati equation*, master's thesis, Nanjing Aeronautical Institute, Campus P.O. Box 245, Nanjing, P.R. China, 1986.
[104] R.B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Mathematics of Computation, 65 (1996), pp. 1213–1230.
[105] E.E. Osborne, *On preconditioning of matrices*, Journal of the ACM, 7 (1960), pp. 338–345.
[106] C.C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, PhD thesis, University of London (UK), 1971.
[107] ———, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 1980 (1980), pp. 235–258.
[108] C.C. Paige and C.F. Van Loan, *A Schur decomposition for Hamiltonian matrices*, Linear Algebra Appl., 41 (1981), pp. 11–32.
[109] B.N. Parlett, *Canonical decomposition of Hessenberg matrices*, Mathematics of Computation, 21 (1967), pp. 223–227.
[110] ———, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.
[111] B.N. Parlett and C. Reinsch, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numerische Mathematik, 13 (1969), pp. 293–304.
[112] B.N. Parlett, D.R. Taylor, and Z.A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Mathematics of Computation, 44 (1985), pp. 105–124.
[113] C. Pester, *COCOS, computation of corner singularities*, tech. report, TU Chemnitz, Preprint SFB393/05-03, 2005.
[114] ———, *Hamiltonian eigenvalue symmetry for quadratic operator eigenvalue problems*, J. Integral Equations Appl., 17 (2005), pp. 71–89.
[115] A.C. Raines and D.S. Watkins, *A class of Hamiltonian-symplectic methods for solving the algebraic riccati equation*, Linear Algebra Appl., 205/206 (1994), pp. 1045–1060.
[116] J. Rosam, *Berechnung der Rissgeometrie bei spröden elastischen Körpern*, Diplomarbeit, TU Chemnitz, 2004.
[117] H. Rutishauser, *Der Quotienten-Differenzen-Algorithmus*, Zeitschrift für angewandte Mathematik und Physik, 5 (1954), pp. 233–251.
[118] ———, *Solution of eigenvalue problems with the LR-transformation*, National Bureau of Standards Applied Mathematics Series, 49 (1958), pp. 47–81.
[119] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

[120] ———, *Numerical methods for large eigenvalue problems: theory and applications*, John Wiley and Sons, New York, 1992.
[121] O. Schenk and K. Gärtner, *Pardiso user guide version 1.2.3*, tech. report, Computer Science Department, University of Basel, Switzerland, 2005.
[122] F. Schmidt, T. Friese, L. Zschiedreich, and P. Deuflhard, *Adaptive multigrid methods for the vectorial Maxwell eigenvalue problem for optical waveguide design*, in Mathematics. Key Technology for the Future, W. Jäger and H.-J. Krebs, eds., Springer-Verlag, 2003, pp. 279–292.
[123] V. Sima, *Algorithms for Linear-Quadratic Optimization*, vol. 200 of Pure and Applied Mathematics, Marcel Dekker, Inc., New York, NY, 1996.
[124] V. Simoncini, *Ritz and pseudo-Ritz values using matrix polynomials*, Linear Algebra Appl., 241-243 (1996), pp. 787–801.
[125] G.L.G. Sleijpen, A.G.L. Booten, D.R. Fokkema, and H.A. Van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.
[126] D.C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[127] ———, *Deflation for implicitly restarted Arnoldi methods*, tech. report, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1998.
[128] ———, *Numerical methods for large eigenvalue problems*, Acta Numerica, (2002), pp. 519–584.
[129] ———, *Passivity preserving model reduction via interpolation of spectral zeros*, Systems & Control Letters, 54 (2005), pp. 347–360.
[130] G.W. Stewart, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.
[131] ———, *Matrix Algorithms, Volume II: Eigensystems*, SIAM , Philadelphia, USA, 2001.
[132] M. Stoll, *Locking und Purging für den Hamiltonischen Lanczos-Prozess*, Diplomarbeit, TU Chemnitz, Fakultät für Mathematik, Germany, 2005.
[133] D.R. Taylor, *Analysis of the look ahead Lanczos algorithm*, PhD thesis, Center for Pure and Applied Mathematics, University of California, Berkley, CA, 1982.
[134] F. Tisseur, *Stability of structured Hamiltonian eigensolvers*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 103–125.
[135] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Review, 43 (2001), pp. 235–286.
[136] C.F. Van Loan, *A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix*, Linear Algebra Appl., 16 (1984), pp. 233–251.
[137] D.S. Watkins, *Understanding the QR algorithm*, SIAM Review, 24 (1982), pp. 427–440.
[138] ———, *Fundamentals of matrix computations*, John Wiley & Sons, Inc., New York, 1991.
[139] ———, *On Hamiltonian and symplectic Lanczos processes*, Linear Algebra Appl., 385 (2004), pp. 23–45.
[140] ———, *On the reduction of a Hamiltonian matrix to Hamiltonian Schur form*, to appear in Electron. Trans. Numer. Anal., (2006).
[141] D.S. Watkins and L. Elsner, *Chasing algorithms for the eigenvalue problem*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 374–384.
[142] ———, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.
[143] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.
[144] J.C. Willems, *Dissipative dynamical systems, part i: General theory*, Arch. Rational Mech. Anal., 45 (1972), pp. 321–351.
[145] N. Wong, V. Balakrishnan, and C.-K. Koh, *Passivity-preserving model reduction via a computationally efficient project-and-balance scheme*, in Proc. Design Automation Conference, San Diego, CA, June 2004, pp. 369–374.
[146] K. Zhou, J.C. Doyle, and K. Glover, *Robust and Optimal Control*, Prentice-Hall, Upper Saddle River, NJ, 1996.

Appendix A. Here we will show that (5.6)

$$\zeta_2''' = c_3^2 c_2 b_2''/s_2$$

holds. We have

$$\begin{aligned}
b_2'' &= b_2' + d_3\nu_2''/c_3 \\
&= (c_2 b_1 + s_2\zeta_2') + d_3(c_2^2\nu_2' - 2c_2 s_2\delta_2' - s_2^2\beta_2')/c_3 \\
&= c_2 b_1 + s_2\zeta_2' + d_3 c_2^2\nu_2'/c_3 - 2d_3 c_2 s_2\delta_2'/c_3 - d_3 s_2^2\beta_2'/c_3 \\
&= c_2 c_1 s_1(\delta_2 - \delta_1) + s_2(c_1^2 - s_1^2)\zeta_2 + s_2 c_1 s_1(\beta_2 - \beta_1) + d_3 c_2^2(c_1^2\nu_2 + s_1^2\nu_1)/c_3 \\
&\quad - 2d_3 c_2 s_2(c_1^2\delta_2 + s_1^2\delta_1)/c_3 - d_3 s_2^2(s_1^2\beta_1 + c_1^2\beta_2 - 2c_1 s_1\zeta_2)/c_3 \\
&= -\delta_1(c_2 c_1 s_1 + 2d_3 c_2 s_2 s_1^2/c_3) + \delta_2(c_2 c_1 s_1 - 2d_3 c_2 s_2 c_1^2/c_3) + \nu_1 d_3 c_2^2 s_1^2/c_3 \\
&\quad + \nu_2 d_3 c_2^2 c_1^2/c_3 - \beta_1(s_2 c_1 s_1 + d_3 s_2^2 s_1^2/c_3) + \beta_2(s_2 c_1 s_1 - d_3 s_2^2 c_1^2/c_3) \\
&\quad + \zeta_2(s_2(c_1^2 - s_1^2) + 2d_3 s_2^2 c_1 s_1/c_3)
\end{aligned}$$

For ease of reference, we will use the following notation for the multiplicative terms

$$b_2'' = -\delta_1\theta_1 + \delta_2\theta_2 + \nu_1\theta_3 + \nu_2\theta_4 - \beta_1\theta_5 + \beta_2\theta_6 + \zeta_2\theta_7$$

Now, let us denote

$$\begin{aligned}
r_1 &= -\delta_1\theta_1 + \delta_2\theta_2 + \nu_1\theta_3 + \nu_2\theta_4, \\
r_2 &= \beta_1\theta_5 + \beta_2\theta_6 + \zeta_2\theta_7,
\end{aligned}$$

then

$$b_2'' = r_1 + r_2.$$

Now let us consider ζ_2'''.

$$\begin{aligned}
\zeta_2''' &= c_3^2\zeta_2'' - c_3 d_3(\widetilde{\delta}_1 + \delta_2''), \\
&= c_3^2(c_2\zeta_2' - s_2 b_1) - c_3 d_3(c_1^2\delta_1 + s_1^2\delta_2 + (c_2^2 - s_2^2)\delta_2' + c_2 s_2(\nu_2' + \beta_2')) \\
&= c_3^2 c_2\zeta_2' - c_3^2 s_2 b_1 - c_3 d_3 c_1^2\delta_1 - c_3 d_3 s_1^2\delta_2 - c_3 d_3(c_2^2 - s_2^2)\delta_2' - c_3 d_3 c_2 s_2\nu_2' \\
&\quad - c_3 d_3 c_2 s_2\beta_2' \\
&= c_3^2 c_2((c_1^2 - s_1^2)\zeta_2 + c_1 s_1(\beta_2 - \beta_1)) - c_3^2 s_2 c_1 s_1(\delta_2 - \delta_1) - c_3 d_3 c_1^2\delta_1 - c_3 d_3 s_1^2\delta_2 \\
&\quad - c_3 d_3(c_2^2 - s_2^2)(c_1^2\delta_2 + s_1^2\delta_1) - c_3 d_3 c_2 s_2(c_1^2\nu_2 + s_1^2\nu_1) \\
&\quad - c_3 d_3 c_2 s_2(s_1^2\beta_1 + c_1^2\beta_2 - 2c_1 s_1\zeta_2) \\
&= \delta_1(c_3^2 s_2 c_1 s_1 - c_3 d_3 c_1^2 - c_3 d_3(c_2^2 - s_2^2)s_1^2) - \delta_2(c_3^2 s_2 c_1 s_1 \\
&\quad + c_3 d_3 s_1^2 + c_3 d_3(c_2^2 - s_2^2)c_1^2) - \nu_1 d_3 c_3 c_2 s_2 s_1^2 - \nu_2 d_3 c_3 c_2 s_2 c_1^2 \\
&\quad - \beta_1(c_3^2 c_2 c_1 s_1 + c_3 d_3 c_2 s_2 s_1^2) + \beta_2(c_3^2 c_2 c_1 s_1 - c_3 d_3 c_2 s_2 c_1^2) \\
&\quad + \zeta_2(c_3^2 c_2(c_1^2 - s_1^2) + 2c_3 d_3 c_2 s_2 c_1 s_1)
\end{aligned}$$

Rewriting the terms multiplied to $\delta_1, \delta_2, \nu_1, \nu_2, \beta_1, \beta_2, \zeta_2$ in terms of $\theta_1, \ldots, \theta_7$ we obtain

$$\begin{aligned}
c_3^2 s_2 c_1 s_1 - c_3 d_3 c_1^2 - c_3 d_3 (c_2^2 - s_2^2) s_1^2 &= c_3^2 s_2 c_1 s_1 - c_3 d_3 c_1^2 - c_3 d_3 (1 - 2 s_2^2) s_1^2 \\
&= c_3^2 s_2 c_1 s_1 - c_3 d_3 c_1^2 - c_3 d_3 s_1^2 + 2 c_3 d_3 s_2^2 s_1^2 \\
&= (c_3^2 s_2 c_1 s_1 + 2 c_3 d_3 s_2^2 s_1^2) - c_3 d_3 \\
&= (c_3^2 s_2 / c_2)(c_2 c_1 s_1 + 2 d_3 c_2 s_2 s_1^2 / c_3) - c_3 d_3 \\
&= c_3^2 s_2 \theta_1 / c_2 - c_3 d_3 \\
-(c_3^2 s_2 c_1 s_1 + c_3 d_3 s_1^2 + c_3 d_3 (c_2^2 - s_2^2) c_1^2) &= -(c_3^2 s_2 c_1 s_1 + c_3 d_3 s_1^2 + c_3 d_3 (1 - 2 s_2^2) c_1^2) \\
&= -(c_3^2 s_2 c_1 s_1 + c_3 d_3 s_1^2 + c_3 d_3 c_1^2 - 2 c_3 d_3 s_2^2 c_1^2) \\
&= -(c_3^2 s_2 c_1 s_1 - 2 c_3 d_3 s_2^2 c_1^2 + c_3 d_3) \\
&= -(c_3^2 s_2 / c_2)(c_2 c_1 s_1 - 2 d_3 c_2 s_2 c_1^2 / c_3) - c_3 d_3 \\
&= -c_3^2 s_2 \theta_2 / c_2 - c_3 d_3 \\
-d_3 c_3 c_2 s_2 s_1^2 &= -(c_3^2 s_2 / c_2)(d_3 c_2^2 s_1^2 / c_3) \\
&= -c_3^2 s_2 \theta_3 / c_2 \\
-d_3 c_3 c_2 s_2 c_1^2 &= -(c_3^2 s_2 / c_2)(d_3 c_2^2 c_1^2 / c_3) \\
&= -c_3^2 s_2 \theta_4 / c_2 \\
-(c_3^2 c_2 c_1 s_1 + c_3 d_3 c_2 s_2 s_1^2) &= -(c_3^2 c_2 / s_2)(s_2 c_1 s_1 + d_3 s_2^2 s_1^2 / c_3) \\
&= -c_3^2 c_2 \theta_5 / s_2 \\
c_3^2 c_2 c_1 s_1 - c_3 d_3 c_2 s_2 c_1^2 &= (c_3^2 c_2 / s_2)(s_2 c_1 s_1 - d_3 s_2^2 c_1^2 / c_3) \\
&= c_3^2 c_2 \theta_6 / s_2 \\
c_3^2 c_2 (c_1^2 - s_1^2) + 2 c_3 d_3 c_2 s_2 c_1 s_1 &= (c_3^2 c_2 / s_2)(s_2 (c_1^2 - s_1^2) + 2 d_3 s_2^2 c_1 s_1 / c_3) \\
&= c_3^2 c_2 \theta_7 / s_2
\end{aligned}$$

and therefore

$$\begin{aligned}
\zeta_2''' &= -c_3 d_3 (\delta_1 + \delta_2) - (c_3^2 s_2 / c_2)(-\delta_1 \theta_1 + \delta_2 \theta_2 + \nu_1 \theta_3 + \nu_2 \theta_4) \\
&\quad + (c_3^2 c_2 / s_2)(-\beta_1 \theta_5 + \beta_2 \theta_6 + \zeta_2 \theta_7) \\
&= -c_3 d_3 (\delta_1 + \delta_2) - (c_3^2 s_2 / c_2) r_1 + (c_3^2 c_2 / s_2) r_2.
\end{aligned}$$

In order to show that

$$\zeta_2''' = c_3^2 c_2 b_2'' / s_2 = c_3^2 c_2 (r_1 + r_2) / s_2,$$

we have to show that

$$-c_3 d_3 (\delta_1 + \delta_2) - (c_3^2 s_2 / c_2) r_1 = c_3^2 c_2 r_1 / s_2.$$

This is equivalent to

$$0 = -c_3 d_3 c_2 s_2 (\delta_1 + \delta_2) - c_3^2 s_2^2 r_1 - c_3^2 c_2^2 r_1 = -c_3 d_3 c_2 s_2 (\delta_1 + \delta_2) - c_3^2 r_1,$$

that is

$$c_3 r_1 = -d_3 c_2 s_2 (\delta_1 + \delta_2).$$

c_2 and s_2 stem from a Givens transformation that eliminates b_2 against b_1, hence

$$\begin{aligned} c_2 &= b_1/\sqrt{t_2} \\ &= c_1 s_1(\delta_2 - \delta_1)/\sqrt{t_2} \\ s_2 &= b_2/\sqrt{t_2} \\ &= c_1 s_1(\nu_2 - \nu_1)/\sqrt{t_2} \end{aligned}$$

where

$$\begin{aligned} t_2 &= b_1^2 + b_2^2 \\ &= c_1^2 s_1^2((\delta_2 - \delta_1)^2 + (\nu_2 - \nu_1)^2). \end{aligned}$$

c_3 and d_3 stems from a Gauss transformation that eliminates b_1' against ν_1', hence for simplicity we can assume that

$$\begin{aligned} c_3 &= \nu_1' = (c_1^2\nu_1 + s_1^2\nu_2) \\ d_3 &= -b_1' = -(c_2 b_1 + s_2 b_2) \\ &= -(c_2 c_1 s_1(\delta_2 - \delta_1) + s_2 c_1 s_1(\nu_2 - \nu_1)) \\ &= -(c_2^2\sqrt{t_2} + s_2^2\sqrt{t_2}) \\ &= -\sqrt{t_2}. \end{aligned}$$

Therefore,

$$\begin{aligned} d_3 s_2 &= -c_1 s_2(\nu_2 - \nu_1), \\ d_3 c_2 &= -c_1 s_1(\delta_2 - \delta_1). \end{aligned}$$

Hence,

$$\begin{aligned} c_3 r_1 &= -\delta_1(c_3 c_2 c_1 s_1 + 2d_3 c_2 s_2 s_1^2) + \delta_2(c_3 c_2 c_1 s_1 - 2d_3 c_2 s_2 c_1^2) + \nu_1 d_3 c_2^2 s_1^2 + \nu_2 d_3 c_2^2 c_1^2 \\ &= c_2\{c_3 c_1 s_1(\delta_2 - \delta_1) - 2d_3 s_2(s_1^2\delta_1 + c_1^2\delta_2) + d_3 c_2(s_1^2\nu_1 + c_1^2\nu_2)\} \\ &= c_2 c_1 s_1\{(c_1^2\nu_1 + s_1^2\nu_2)(\delta_2 - \delta_1) + 2(\nu_2 - \nu_1)(s_1^2\delta_1 + c_1^2\delta_2) - (\delta_2 - \delta_1)(s_1^2\nu_1 + c_1^2\nu_2)\} \\ &= c_2 c_1 s_1\{(\delta_2 - \delta_1)(c_1^2\nu_1 + s_1^2\nu_2 - s_1^2\nu_1 - c_1^2\nu_2) + 2(\nu_2 - \nu_1)(s_1^2\delta_1 + c_1^2\delta_2)\} \\ &= c_2 c_1 s_1\{(\delta_2 - \delta_1)(c_1^2(\nu_1 - \nu_2) + s_1^2(\nu_2 - \nu_1)) + 2(\nu_2 - \nu_1)(s_1^2\delta_1 + c_1^2\delta_2)\} \\ &= c_2 c_1 s_1\{(\delta_2 - \delta_1)(\nu_2 - \nu_1)(s_1^2 - c_1^2) + 2(\nu_2 - \nu_1)(s_1^2\delta_1 + c_1^2\delta_2)\} \\ &= c_2 c_1 s_1(\nu_2 - \nu_1)\{(\delta_2 - \delta_1)(s_1^2 - c_1^2) + 2(s_1^2\delta_1 + c_1^2\delta_2)\} \\ &= c_2 c_1 s_1(\nu_2 - \nu_1)\{s_1^2\delta_2 - s_1^2\delta_1 - c_1^2\delta_2 + c_1^2\delta_1 + 2(s_1^2\delta_1 + c_1^2\delta_2)\} \\ &= c_2 c_1 s_1(\nu_2 - \nu_1)\{s_1^2\delta_2 - c_1^2\delta_2 + 2c_1^2\delta_2 - s_1^2\delta_1 + c_1^2\delta_1 + 2s_1^2\delta_1\} \\ &= c_2 c_1 s_1(\nu_2 - \nu_1)(\delta_2 + \delta_1) \\ &= -d_3 c_2 s_2(\delta_1 + \delta_2). \end{aligned}$$

Hence, (5.6) holds.

Appendix B.

```
function [d,b,z,v,S,error,condmax] = ...
      param_sr_implicit_single(d,b,z,v,shift,S,n,badcond)

%  Computes a single shifted implicit SR step for a parameterized
%  J-Hessenberg matrix. Given a Hamiltonian matrix
%  H = [diag(d) diag(b)+diag(z(2:n),1)+diag(z(2:n),-1); diag(v) -diag(d)]
%  and a real shift, the SR decomposition of H - shift*I is SR with a symplectic
%  matrix SS and an upper J-triangular matrix R.
%  An SR step computes HH = inv(SS)*H*SS = R*SS + shift*I.
%  Here, HH = inv(SS)*H*SS is computed implicitly. H is never used explicitly, all
%  computations are done on the parameters which determine H, that is the parameters
%  which determine HH are computed.
%
%  If the condition number of any of the Gaussian elimination matrices
%  is larger than the given tolerance badcond, error is set to 1.
%  Otherwise error is set to 0 and condmax is set to the maximum of
%  the condition numbers of the previously used Gaussian elimination matrices
%  and the condition number of the current elimination matrix.
%
% on input
%       d       - delta(1:n) which determines H as described above
%       b       - beta(1:n) which determines H as described above
%       z       - zeta(2:n) which determines H as described above
%       v       - nu(1:n) which determines H as described above
%       shift   - real shift to be used in the implicit SR step
%       S       - symplectic transformation matrix used so far
%       n       - size of d, b and v, S is 2n-x-2n
%       badcond - tolerance for maximal admissible condition number
%                    of Gaussian elimination matrices
%
%
% on output
%    d          - delta(1:n) which determines HH as described above
%    b          - beta(1:n) which determines HH as described above
%    z          - zeta(2:n) which determines HH as described above
%    v          - nu(1:n) which determines HH as described above
%    S          - updated symplectic transformation matrix
%    error      - 0 if there was no problem with the Gaussian elimination,
%                 1 otherwise
%    condmax    - maximal condition number of all symplectic Gauss
%                   transformation used
%
% Reference:
% H. Fassbender: The parameterized Hamiltonian SR algorithm
% Preprint 2005
%
%
% H. Fassbender, 12/2005 MATLAB 7.1.0.246 (R14) Service Pack3
condmax = 1;
%
%first step, create bulge
%
[c,s] = givens(d(1)-shift,v(1));
```

```
c2 = c*c;
s2 = s*s;
cs = c*s;
dneu = (c2-s2)*d(1) +cs*(v(1)+b(1));
vneu = c2*v(1)-s2*b(1)-2*cs*d(1);
bneu = c2*b(1)-s2*v(1)-2*cs*d(1);
zneu = c*z(2);
bulge = s*z(2);
d(1) = dneu;
v(1) = vneu;
b(1) = bneu;
z(2) = zneu;
S(1:2*n,[1 n+1]) = S(1:2*n,[1 n+1])*[c -s; s c];
%
%  chase the bulge
%
for j = 1:n-1
    %
    % Gaussian elimination to delete bulge in (j+1,j),
    % introduces a new one in (j,j+1)
    %
    [c,g,Lcond,error] = gauss1(bulge,v(j),condmax,badcond);
    condmax = max(condmax,Lcond);
    if condmax < badcond
        c2 = c*c;
        g2 = g*g;
        cg = c*g;
        vjneu = v(j)/c2;
        vjp1neu = v(j+1)/c2;
        bjneu = -v(j+1)*g2+c2*b(j);
        bjp1neu = c2*b(j+1)-cg*bulge;
        zjneu = c*z(j);
        zjp1neu = -cg*(d(j)+d(j+1))+c2*z(j+1);
        bulge = g*v(j+1)/c;
        v(j) = vjneu;
        v(j+1) = vjp1neu;
        b(j) = bjneu;
        b(j+1) = bjp1neu;
        z(j) = zjneu;
        z(j+1) = zjp1neu;
        if j < n-1
            zjp2neu = c*z(j+2);
            z(j+2) = zjp2neu;
        end
        S(1:2*n,[j j+1 n+j n+j+1]) =  S(1:2*n,[j j+1 n+j n+j+1]) * ...
                [1/c 0 0 -g; 0 1/c -g 0; 0 0 c 0; 0 0 0 c];
        %
        % Givens elimination to delete bulge in (j,j+1),
        % introduces a new one in (j+2,j+1)
        %
        [c,s] = givens(z(j+1),-bulge);
        c2 = c*c;
        s2 = s*s;
        cs = c*s;
```

```
        djp1neu = c2*d(j+1)+cs*(v(j+1)+b(j+1))-s2*d(j+1);
        vjp1neu = c2*v(j+1)-2*cs*d(j+1)-s2*b(j+1);
        zjp1neu = c*z(j+1)-s*bulge;
        bjp1neu = c2*b(j+1)-2*cs*d(j+1)-s2*v(j+1);
        bulge = s*z(j+2);
        d(j+1) = djp1neu;
        v(j+1) = vjp1neu;
        z(j+1) = zjp1neu;
        b(j+1) = bjp1neu;
        if j < n-1
            zjp2neu = c*z(j+2);
            z(j+2) = zjp2neu;
        end
        S(1:2*n,[j+1 n+j+1]) = S(1:2*n,[j+1 n+j+1])*[c -s; s c];
    else
        error = 1;
        return
    end
end
```

```
function [d,b,z,v,S,error,condmax] = ...
   param_sr_implicit_double(d,b,z,v,shift,S,n,badcond)

%  Computes a double shifted implicit SR step for a parameterized
%  Hamiltonian J-Hessenberg matrix. Given a Hamiltonian matrix
%  H = [diag(d) diag(b)+diag(z(2:n),1)+diag(z(2:n),-1); diag(v) -diag(d)]
%  and a real or  a purely imaginary shift, the SR decomposition of
%  (H - shift*I)(H + shift*I) is SR with a symplectic
%  matrix SS and an upper J-triangular matrix R.
%  An SR step computes HH = inv(SS)*H*SS.
%  Here, HH = inv(SS)*H*SS is computed implicitly. H is never used explicitly, all
%  computations are done on the parameters which determine H, that is the parameters
%  which determine HH are computed.
%  In case shift == Inf, a Rayleigh-quotient like shift strategy is used.
%
%  If the condition number of any of the Gaussian elimination matrices
%  is larger than the given tolerance badcond, error is set to 1.
%  Otherwise error is set to 0 and condmax is set to the maximum of
%  the condition numbers of the previously used Gaussian elimination matrices
%  and the condition number of the current elimination matrix.
%
% on input
%       d       - delta(1:n) which determines H as described above
%       b       - beta(1:n) which determines H as described above
%       z       - zeta(2:n) which determines H as described above
%       v       - nu(1:n) which determines H as described above
%       shift   - real shift to be used in the implicit SR step
%                 if shift == Inf, a Rayleigh-quotient like shift is used
%       S       - symplectic transformation matrix used so far (might be
%                 nonsquare, if used inside SR iteration when deflation has
%                 taken place)
%       n       - size of d, b and v, S is sn-x-2n
%       badcond - tolerance for maximal admissible condition number
%                   of Gaussian elimination matrices
%
%
% on output
%    d          - delta(1:n) which determines HH as described above
%    b          - beta(1:n) which determines HH as described above
%    z          - zeta(2:n) which determines HH as described above
%    v          - nu(1:n) which determines HH as described above
%    S          - updated symplectic transformation matrix
%    error      - 0 if there was no problem with the Gaussian elimination,
%                 1 otherwise
%    condmax    - maximal condition number of all symplectic Gauss
%                  transformation used
%
% Reference:
% H. Fassbender: The parameterized Hamiltonian SR algorithm
% Preprint 2006
%
%
% H. Fassbender, 01/2006 MATLAB 7.1.0.246 (R14) Service Pack3
condmax = 1; sn = size(S,1);
```

```
%
%first step, create bulge
%
if shift == Inf
    [c,s]  = givens(d(1)*d(1)+v(1)*b(1)-d(n)^2-b(n)*v(n),v(1)*z(2));
else
    [c,s]  = givens(d(1)*d(1)+v(1)*b(1)-shift*shift,v(1)*z(2));
end
c2     = c*c;
s2     = s*s;
cs     = c*s;
d1neu  = c2*d(1) + s2*d(2);
d2neu  = s2*d(1) + c2*d(2);
v1neu  = c2*v(1) + s2*v(2);
v2neu  = s2*v(1) + c2*v(2);
b1neu  = c2*b(1) + s2*b(2) + 2*cs*z(2);
b2neu  = s2*b(1) + c2*b(2) - 2*cs*z(2);
z2neu  = (c2-s2)*z(2) + cs*(b(2)-b(1));
bulge1 = cs*(d(2)-d(1));
bulge2 = cs*(v(2)-v(1));
d(1)   = d1neu;
d(2)   = d2neu;
v(1)   = v1neu;
v(2)   = v2neu;
b(1)   = b1neu;
b(2)   = b2neu;
z(2)   = z2neu;
if n > 2
    z3neu  = c*z(3);
    bulge3 = s*z(3);
    z(3)   = z3neu;
end
S(1:sn,[1 2])     = S(1:sn,[1 2])*[c -s; s c];
S(1:sn,[n+1 n+2]) = S(1:sn,[n+1 n+2])*[c -s; s c];
%
%  chase the bulge
%
for j = 1:n-1
    %
    % Givens elimination to delete bulge in (n+j+1,n+j),
    %
    [c,s]     = givens(bulge1,bulge2);
    c2        = c*c;
    s2        = s*s;
    cs        = c*s;
    djp1neu   = (c2-s2)*d(j+1) + cs*(v(j+1)+b(j+1));
    vjp1neu   = c2*v(j+1) -2*cs*d(j+1) - s2*b(j+1);
    zjp1neu   = c*z(j+1) - s*bulge1;
    bjp1neu   = c2*b(j+1) - 2*cs*d(j+1) - s2*v(j+1);
    bulge1neu = c*bulge1 + s*bulge2;
    bulge2neu =  c*bulge1 + s*z(j+1);
    d(j+1)    = djp1neu;
    v(j+1)    = vjp1neu;
    z(j+1)    = zjp1neu;
```

```
b(j+1)    = bjp1neu;
bulge1    = bulge1neu;
bulge2    = bulge2neu;
if (j < n-1) & (n > 2)
    zjp2neu = c*z(j+2);
    bulge4  = s*z(j+2);
    z(j+2)  = zjp2neu;
end
S(1:sn,[j+1 n+j+1]) = S(1:sn,[j+1 n+j+1])*[c -s; s c];
%
% Gauss elimination to delete bulge in (j+1,j),
% no new entry created
%
[c,g,Lcond,error] = gauss1(bulge1,v(j),condmax,badcond);
condmax           = max(condmax,Lcond);
if condmax < badcond
    c2      = c*c;
    g2      = g*g;
    cg      = c*g;
    vjneu   = v(j)/c2;
    vjp1neu = v(j+1)/c2;
    bjneu   = c2*b(j) - 2*cg*bulge2 - g2*v(j+1);
    bjp1neu = c2*b(j+1) - cg*bulge1;
    zjneu   = c*z(j);
    zjp1neu = c2*z(j+1) - cg*(d(j)+d(j+1));
    bulge1  = c*bulge1 + g*v(1);
    bulge2  = bulge2 + g*v(j+1)/c;
    v(j)    = vjneu;
    v(j+1)  = vjp1neu;
    b(j)    = bjneu;
    b(j+1)  = bjp1neu;
    z(j)    = zjneu;
    z(j+1)  = zjp1neu;
    if (j < n-1) & (n > 2)
        zjp2neu = c*z(j+2);
        z(j+2)  = zjp2neu;
        bulge3  =  c*bulge3 - g*bulge4;
        bulge4  = bulge4/c;
    end
    S(1:sn,[j j+1 n+j n+j+1]) = S(1:sn,[j j+1 n+j n+j+1]) * ...
        [1/c 0 0 -g; 0 1/c -g 0; 0 0 c 0; 0 0 0 c];
    %
    % Givens elimination to delete bulge in (n+j+1,n+j)
    % no new entry created
    %
    [c,s]   = givens(z(j+1),-bulge2);
    c2      = c*c;
    s2      = s*s;
    cs      = c*s;
    djp1neu = c2*d(j+1) + cs*(v(j+1)+b(j+1)) - s2*d(j+1);
    vjp1neu = c2*v(j+1) -2*cs*d(j+1) - s2*b(j+1);
    zjp1neu = -s*bulge2 + c*z(j+1);
    bjp1neu = c2*b(j+1) - 2*cs* d(j+1) - s2*v(j+1);
    d(j+1)  = djp1neu;
```

```
        v(j+1)  = vjp1neu;
        z(j+1)  = zjp1neu;
        b(j+1)  = bjp1neu;
        if (j < n-1) & (n > 2)
            zjp2neu = -s*bulge4 + c*z(j+2);
            bulge4  = c*bulge4 + s*z(j+2);
            z(j+2)  = zjp2neu;
        end
        S(1:sn,[j+1 n+j+1]) = S(1:sn,[j+1 n+j+1])*[c -s;s c];
        %
        % Givens Type II to delete bulge in (j+3,n+j)
        %
        if (j < n-1) & (n > 2)
            [c,s]   = givens(z(j+1),bulge3);
            c2      = c*c;
            s2      = s*s;
            cs      = c*s;
            djp1neu = c2*d(j+1) + cs*bulge4 + s2*d(j+2);
            djp2neu = s2*d(j+1) -cs*bulge4 + c2*d(j+2);
            vjp1neu = c2*v(j+1) + s2*v(j+2);
            vjp2neu = c2*v(j+2) + s2*v(j+1);
            zjp1neu = c*z(j+1) + s*bulge3;
            zjp2neu = c2*z(j+2) +cs*(b(j+2)-b(j+1)) - s2*z(j+2);
            bjp1neu = c2*b(j+1) + 2*cs*z(j+2) + s2*b(j+2);
            bjp2neu = s2*b(j+1) - 2*cs*z(j+2) + c2*b(j+2);
            bulge1  = c2*bulge4 + cs*(d(j+2)-d(j+1));
            bulge2  = cs*(v(j+2)-v(j+1));
            d(j+1)  = djp1neu;
            d(j+2)  = djp2neu;
            v(j+1)  = vjp1neu;
            v(j+2)  = vjp2neu;
            z(j+1)  = zjp1neu;
            z(j+2)  = zjp2neu;
            b(j+1)  = bjp1neu;
            b(j+2)  = bjp2neu;
            if j < n-2
                zjp3neu = c*z(j+3);
                bulge3  = s*z(j+3);
                z(j+3)  = zjp3neu;
            end
            S(1:sn,[j+1 j+2])     = S(1:sn,[j+1 j+2])*[c -s; s c];
            S(1:sn,[n+j+1 n+j+2]) = S(1:sn,[n+j+1 n+j+2])*[c -s; s c];
        end
    else
        error = 1;
        return
    end
end
```

```
function [d,b,z,v,S,error,condmax] = ...
    param_sr_implicit_quadruple(d,b,z,v,shift,S,n,badcond)

%  Computes a quadruple shifted implicit SR step for a parameterized
%  Hamiltonian J-Hessenberg matrix. Given a 2n-x-2n Hamiltonian matrix
%  H = [diag(d) diag(b)+diag(z(2:n),1)+diag(z(2:n),-1); diag(v) -diag(d)]
%  and a quadruple of complex values (shift, -shift, conj(shift), -conj(shift)),
%  the SR decomposition of
%  HH = (H - shift I)(H + shift I)(H - conj(shift) I)(H + conj(shift) I)
%  is SR with a symplectic matrix S and an upper J-triangular matrix R.
%  An SR step computes HH = inv(SS)*H*SS.
%  Here, HH = inv(SS)*H*SS is computed implicitly. H is never used explicitly, all
%  computations are done on the parameters which determine H, that is the parameters
%  which determine HH are computed.
%  In case shift == Inf, a Rayleigh-quotient like shift strategy is used.
%
%  If the condition number of any of the Gaussian elimination matrices
%  is larger than the given tolerance badcond, error is set to 1.
%  Otherwise error is set to 0 and condmax is set to the maximum of
%  the condition numbers of the previously used Gaussian elimination matrices
%  and the condition number of the current elimination matrix.
%
% on input
%       d       - delta(1:n) which determines H as described above
%       b       - beta(1:n) which determines H as described above
%       z       - zeta(2:n) which determines H as described above
%       v       - nu(1:n) which determines H as described above
%       shift   - complex shift to be used in the implicit SR step
%                 if shift == Inf, a Rayleigh-quotient like shift is used
%       S       - symplectic transformation matrix used so far (might be
%                 nonsquare, if used inside SR iteration when deflation has
%                 taken place)
%       n       - size of d, b and v, S is sn-x-2n, n > 3
%       badcond - tolerance for maximal admissible condition number
%                   of Gaussian elimination matrices
%
%
% on output
%    d          - delta(1:n) which determines HH as described above
%    b          - beta(1:n) which determines HH as described above
%    z          - zeta(2:n) which determines HH as described above
%    v          - nu(1:n) which determines HH as described above
%    S          - updated symplectic transformation matrix
%    error      - 0 if there was no problem with the Gaussian elimination,
%                 1 otherwise
%    condmax    - maximal condition number of all symplectic Gauss
%                  transformation used
%
% Reference:
% H. Fassbender: The parameterized Hamiltonian SR algorithm
% Preprint 2006
%
%
% H. Fassbender, 01/2006 MATLAB 7.1.0.246 (R14) Service Pack3
```

```
if n < 4
    [d,b,z,v,S,error,condmax] = ...
        param_sr_implicit_quadruple_6(d,b,z,v,shift,S,badcond);
    return
end
sn = size(S,1);
condmax = 1;
error   = 0;
%
% first column of ...
%
ha = d(1)^2+v(1)*b(1);
x1 = ha^2 + v(1)*v(2)*z(2)^2;
if shift == Inf
    h1 = d(n-1)^2 +v(n-1)*b(n-1);
    h2 = d(n)^2 + v(n)*b(n);
    h = h1 + h2;
    x1 = x1 - h*ha + h1*h2 - v(n-1)*v(n)*z(n)^2;
else
    h  = real(2*(real(shift)^2 - imag(shift)^2));
    x1 = x1 - h*ha + abs(shift)^4;
end;
x2 = v(1)*z(2)*(ha + d(2)^2 + v(2)*b(2) - h);
x3 = v(1)*v(2)*z(2)*z(3);
%
% create bulge, first part
%
[c,s]  = givens(x2,x3);
c2     = c*c;
s2     = s*s;
cs     = c*s;
d2neu  = c2*d(2) + s2*d(3);
d3neu  = s2*d(2) + c2*d(3);
v2neu  = c2*v(2) + s2*v(3);
v3neu  = s2*v(2) + c2*v(3);
z2neu  = c*z(2);
z3neu  = c2*z(3) + cs*(b(3)-b(2)) - s2*z(3);
z4neu  = c*z(4);
b2neu = c2*b(2) + s2*b(3) + 2*cs*z(3);
b3neu  = s2*b(2) + c2*b(3) - 2*cs*z(3);
bulge1 = cs*(d(3)-d(2));
bulge2 = cs*(v(3)-v(2));
bulge3 = -s*z(2);
bulge4 = s*z(4);
d(2)   = d2neu;
d(3)   = d3neu;
v(2)   = v2neu;
v(3)   = v3neu;
b(2)   = b2neu;
b(3)   = b3neu;
z(2)   = z2neu;
z(3)   = z3neu;
z(4)   = z4neu;
S(1:sn,[2 3]) = S(1:sn,[2 3])*[c -s; s c];
```

```
S(1:sn,[n+2 n+3]) = S(1:sn,[n+2 n+3])*[c -s; s c];
%
% create bulge, second part
%
[c,s] = givens(x1,c*x2+s*x3);
c2 = c*c;
s2 = s*s;
cs = c*s;
d1neu     = c2*d(1) + s2*d(2);
d2neu     = c2*d(2) + s2*d(1);
v1neu     = c2*v(1) + s2*v(2);
v2neu     = c2*v(2) + s2*v(1);
b1neu     = c2*b(1) + 2*cs*z(2) + s2*b(2);
b2neu     = c2*b(2) - 2*cs*z(2) + s2*b(1);
z2neu     = c2*z(2) + cs*(b(2)-b(1)) - s2*z(2);
z3neu     = c*z(3) - s*bulge3;
bulge1neu = cs*(d(2)-d(1));
bulge2neu = s*bulge1;
bulge3neu = c*bulge1;
bulge4neu = cs*(v(2)-v(1));
bulge5    = s*bulge2;
bulge6    = c*bulge2;
bulge7    = c*bulge3 + s*z(3);
bulge8    = s*bulge4;
bulge9    = c*bulge4;
d(1)      = d1neu;
d(2)      = d2neu;
v(1)      = v1neu;
v(2)      = v2neu;
b(1)      = b1neu;
b(2)      = b2neu;
z(2)      = z2neu;
z(3)      = z3neu;
bulge1    = bulge1neu;
bulge2    = bulge2neu;
bulge3    = bulge3neu;
bulge4    = bulge4neu;
bulge10   = bulge1;
bulge11   = bulge2;
bulge12   = bulge3;
S(1:sn,[1 2]) = S(1:sn,[1 2])*[c -s; s c];
S(1:sn,[n+1 n+2]) = S(1:sn,[n+1 n+2])*[c -s; s c];

for j = 1:n-1,
    %
    %   Chase the bulge
    %
    if j < n-1
        %
        % Givens elimination to delete bulge in (n+2+1,j),
        %
        [c,s]      = givens(bulge2,bulge5);
        c2         = c*c;
        s2         = s*s;
```

```
    cs         = c*s;
    djp2neu    = (c2-s2)*d(j+2) + cs*(v(j+2)+b(j+2));
    vjp2neu    = c2*v(j+2) -2*cs*d(j+2) - s2*b(j+2);
    zjp2neu    = c*z(j+2) - s*bulge12;
    bjp2neu    = c2*b(j+2) - 2*cs*d(j+2) - s2*v(j+2);
    bulge2neu  = c*bulge2 + s*bulge5;
    bulge3neu  = c*bulge3 + s*bulge6;
    bulge6neu  = c*bulge6 - s*bulge3;
    bulge7neu  = c*bulge7 - s*bulge11;
    bulge11neu = c*bulge11 + s*bulge7;
    bulge12neu = c*bulge12 + s*z(j+2);
    d(j+2)     = djp2neu;
    v(j+2)     = vjp2neu;
    z(j+2)     = zjp2neu;
    b(j+2)     = bjp2neu;
    bulge2     = bulge2neu;
    bulge3     = bulge3neu;
    bulge6     = bulge6neu;
    bulge7     = bulge7neu;
    bulge11    = bulge11neu;
    bulge12    = bulge12neu;
    if j < n-2
        zjp3neu    = c*z(j+3);
        bulgex     = s*z(j+3);
        z(j+3)     = zjp3neu;
    end
    S(1:sn,[j+2 n+j+2]) = S(1:sn,[j+2 n+j+2])*[c -s; s c];
end
%
% Givens elimination to delete bulge in (n+j+1,j),
%
[c,s]      = givens(bulge1,bulge4);
c2         = c*c;
s2         = s*s;
cs         = c*s;
djp1neu    = (c2-s2)*d(j+1) + cs*(v(j+1)+b(j+1));
vjp1neu    = c2*v(j+1)-2*cs*d(j+1)-s2*b(j+1);
bjp1neu    = c2*b(j+1) - 2*cs*d(j+1) - s2*v(j+1);
zjp1neu    = c*z(j+1) - s*bulge10;
bulge1neu  = c*bulge1 + s*bulge4;
bulge4neu  = c*bulge6 - s*bulge12;
bulge10neu = c*bulge10 + s*z(j+1);
bulge12neu = c*bulge12 + s*bulge6;
d(j+1)     = djp1neu;
v(j+1)     = vjp1neu;
b(j+1)     = bjp1neu;
z(j+1)     = zjp1neu;
bulge1     = bulge1neu;
bulge4     = bulge4neu;
bulge10    = bulge10neu;
bulge12    = bulge12neu;
if j < n-2
    bulge9neu  = c*bulge9;
    bulgey     = s*bulge9;
```

```
    bulge9     = bulge9neu;
end
if j < n-1
    zjp2neu    = c*z(j+2) -s*bulge3;
    bulge3neu  = c*bulge3 + s*z(j+2);
    z(j+2)     = zjp2neu;
    bulge3     = bulge3neu;
end
S(1:sn,[j+1 n+j+1]) = S(1:sn,[j+1 n+j+1])*[c -s; s c];
if j < n-1
    %
    % Givens elimination type II to delete bulge in (j+2,j),
    %
    [c,s]      = givens(bulge1,bulge2);
    c2         = c*c;
    s2         = s*s;
    cs         = c*s;
    djp1neu    = c2*d(j+1) + cs*(bulge3+bulge12) + s2*d(j+2);
    djp2neu    = c2*d(j+2) - cs*(bulge3+bulge12) + s2*d(j+1);
    vjp1neu    = c2*v(j+1) + 2*cs*bulge4 + s2*v(j+2);
    vjp2neu    = c2*v(j+2) - 2*cs*bulge4 + s2*v(j+1);
    bjp1neu    = c2*b(j+1) + 2*cs*z(j+2) + s2*b(j+2);
    bjp2neu    = c2*b(j+2) - 2*cs*z(j+2) + s2*b(j+1);
    zjp1neu    = c*z(j+1) + s*bulge7;
    zjp2neu    = (c2-s2)*z(j+2) + cs*(b(j+2)-b(j+1));
    bulge1neu  = c*bulge1 + s*bulge2;
    bulge3neu  = c2*bulge3 + cs*(d(j+2)-d(j+1)) -s2*bulge12;
    bulge4neu  = (c2-s2)*bulge4 + cs*(v(j+2)-v(j+1));
    bulge7neu  = c*bulge7 - s*z(j+1);
    bulge10neu = c*bulge10 + s*bulge11;
    bulge11neu = c*bulge11 - s*bulge10;
    bulge12neu = c2*bulge12 + cs*(d(j+2)-d(j+1)) -s2*bulge3;
    d(j+1)     = djp1neu;
    d(j+2)     = djp2neu;
    v(j+1)     = vjp1neu;
    v(j+2)     = vjp2neu;
    b(j+1)     = bjp1neu;
    b(j+2)     = bjp2neu;
    z(j+1)     = zjp1neu;
    z(j+2)     = zjp2neu;
    bulge1     = bulge1neu;
    bulge3     = bulge3neu;
    bulge4     = bulge4neu;
    bulge7     = bulge7neu;
    bulge10    = bulge10neu;
    bulge11    = bulge11neu;
    bulge12    = bulge12neu;
    if j < n-2
        zjp3neu    = c*z(j+3) - s*bulge9;
        bulge9neu  = c*bulge9 + s*z(j+3);
        bulgexneu  = c*bulgex - s*bulgey;
        bulgeyneu  = c*bulgey + s*bulgex;
        z(j+3)     = zjp3neu;
        bulge9     = bulge9neu;
```

```
        bulgex     = bulgexneu;
        bulgey     = bulgeyneu;
    end
    S(1:sn,[j+1 j+2])     = S(1:sn,[j+1 j+2])*[c -s; s c];
    S(1:sn,[n+j+1 n+j+2]) = S(1:sn,[n+j+1 n+j+2])*[c -s; s c];
end
%
% Gauss elimination to delete bulge in (j+1,j),
% no new entry created
%
[c,g,Lcond,error] = gauss1(bulge1,v(j),condmax,badcond);
condmax = max(condmax,Lcond);
if condmax < badcond
    c2         = c*c;
    g2         = g*g;
    cg         = c*g;
    vjneu      = v(j)/c2;
    vjp1neu    = v(j+1)/c2;
    bjneu      = c2*b(j) - 2*cg*bulge10 - g2*v(j+1);
    bjp1neu    = c2*b(j+1) - 2*cg*bulge1 - g2*v(j);
    zjp1neu    = c2*z(j+1) - cg*(d(j)+d(j+1));
    zjneu      = c*z(j);
    bulge1neu  = bulge3/c;
    bulge4neu  = bulge4/c;
    bulge7neu  = c*bulge7 - g*bulge3;
    bulge10neu = bulge10 + g*v(j+1)/c;
    bulge11neu = c*bulge11 + g*bulge4 ;
    bulge12neu = c*bulge12;
    v(j)       = vjneu;
    v(j+1)     = vjp1neu;
    b(j)       = bjneu;
    b(j+1)     = bjp1neu;
    z(j)       = zjneu;
    z(j+1)     = zjp1neu;
    bulge1     = bulge1neu;
    bulge4     = bulge4neu;
    bulge7     = bulge7neu;
    bulge10    = bulge10neu;
    bulge11    = bulge11neu;
    bulge12    = bulge12neu;
    if j < n-2
        bulge2neu  = bulgey/c;
        bulge3neu  = bulgex;
        bulge8neu  = c*bulge8 - g*bulgey;
        bulge9neu  = c*bulge9;
        bulge2     = bulge2neu;
        bulge3     = bulge3neu;
        bulge8     = bulge8neu;
        bulge9     = bulge9neu;
    end
    if j < n-1
        zjp2neu    = c*z(j+2);
        z(j+2)     = zjp2neu;
    end
```

```
S(1:sn,[j j+1 n+j n+j+1]) = S(1:sn,[j j+1 n+j n+j+1]) * ...
      [1/c 0 0 -g; 0 1/c -g 0; 0 0 c 0; 0 0 0 c];
if j < n-1
    %
    % Givens elimination to delete bulge in (n+j+2,n+j),
    %
    [c,s]      = givens(bulge7,-bulge11);
    c2         = c*c;
    s2         = s*s;
    cs         = c*s;
    djp2neu    = (c2-s2)*d(j+2) + cs*(v(j+2)+b(j+2));
    vjp2neu    = c2*v(j+2) - 2*cs*d(j+2) - s2*b(j+2);
    bjp2neu    = c2*b(j+2) - 2*cs*d(j+2) - s2*v(j+2);
    zjp2neu    = c*z(j+2) - s*bulge12;
    bulge1neu  = c*bulge1 + s*bulge4;
    bulge4neu  = c*bulge4 - s*bulge1;
    bulge7neu  = c*bulge7 - s*bulge11;
    bulge12neu = c*bulge12 + s*z(j+2);
    d(j+2)     = djp2neu;
    v(j+2)     = vjp2neu;
    b(j+2)     = bjp2neu;
    z(j+2)     = zjp2neu;
    bulge1     = bulge1neu;
    bulge4     = bulge4neu;
    bulge7     = bulge7neu;
    bulge12    = bulge12neu;
    if j < n-2
        zjp3neu    = c*z(j+3) - s*bulge3;
        bulge3neu  = c*bulge3 + s*z(j+3);
        z(j+3)     = zjp3neu;
        bulge3     = bulge3neu;
    end
    S(1:sn,[j+2 n+j+2]) = S(1:sn,[j+2 n+j+2])*[c -s; s c];
end
%
% Givens elimination to delete bulge in (n+j+1,n+j),
%
[c,s]      = givens(z(j+1),-bulge10);
c2         = c*c;
s2         = s*s;
cs         = c*s;
djp1neu    = (c2-s2)*d(j+1) + cs*(v(j+1)+b(j+1));
vjp1neu    = c2*v(j+1) - 2*cs*d(j+1) - s2*b(j+1);
bjp1neu    = c2*b(j+1) - 2*cs*d(j+1) - s2*v(j+1);
zjp1neu    = c*z(j+1) - s*bulge10;
bulge4neu  = c*bulge4 - s*bulge12;
bulge12neu = c*bulge12 + s*bulge4;
d(j+1)     = djp1neu;
v(j+1)     = vjp1neu;
b(j+1)     = bjp1neu;
z(j+1)     = zjp1neu;
bulge4     = bulge4neu;
bulge12    = bulge12neu;
if j < n-2
```

```
    bulge2neu  = c*bulge2 + s*bulge9;
    bulge9neu  = c*bulge9 - s*bulge2;
    bulge2     = bulge2neu;
    bulge9     = bulge9neu;
end
if j < n-1
    zjp2neu    = c*z(j+2) - s*bulge1;
    bulge1neu  = c*bulge1 + s*z(j+2);
    z(j+2)     = zjp2neu;
    bulge1     = bulge1neu;
end
S(1:sn,[j+1 n+j+1]) = S(1:sn,[j+1 n+j+1])*[c -s; s c];
if j < n-2
    %
    % Givens elimination type II to delete bulge in (j+3,n+j),
    %
    [c,s]      = givens(bulge7,bulge8);
    c2         = c*c;
    s2         = s*s;
    cs         = c*s;
    djp2neu    = c2*d(j+2) + cs*bulge3 + s2*d(j+3);
    djp3neu    = c2*d(j+3) - cs*bulge3 + s2*d(j+2);
    vjp2neu    = c2*v(j+2) + s2*v(j+3);
    vjp3neu    = c2*v(j+3) + s2*v(j+2);
    bjp2neu    = c2*b(j+2) + 2*cs*z(j+3) + s2*b(j+3);
    bjp3neu    = c2*b(j+3) - 2*cs*z(j+3) + s2*b(j+2);
    zjp2neu    = c*z(j+2) + s*bulge9;
    zjp3neu    = (c2-s2)*z(j+3) + cs*(b(j+3) -b(j+2));
    bulge1neu  = c*bulge1 + s*bulge2;
    bulge2neu  = c*bulge2 - s*bulge1;
    bulge3neu  = c2*bulge3 + cs*(d(j+3)-d(j+2));
    bulge4neu  = c*bulge4;
    bulge5neu  = -s*bulge4;
    bulge6neu  = cs*(v(j+3) -v(j+2));
    bulge7neu  = c*bulge7 + s*bulge8;
    bulge9neu  = c*bulge9 - s*z(j+2);
    bulge10neu = c*bulge12;
    bulge11neu = -s*bulge12;
    bulge12neu = cs*(d(j+3)-d(j+2))-s^2*bulge3;
    d(j+2)     = djp2neu;
    d(j+3)     = djp3neu;
    v(j+2)     = vjp2neu;
    v(j+3)     = vjp3neu;
    b(j+2)     = bjp2neu;
    b(j+3)     = bjp3neu;
    z(j+2)     = zjp2neu;
    z(j+3)     = zjp3neu;
    bulge1     = bulge1neu;
    bulge2     = bulge2neu;
    bulge3     = bulge3neu;
    bulge4     = bulge4neu;
    bulge5     = bulge5neu;
    bulge6     = bulge6neu;
    bulge7     = bulge7neu;
```

```
    bulge9     = bulge9neu;
    bulge10    = bulge10neu;
    bulge11    = bulge11neu;
    bulge12    = bulge12neu;
    if j < n-3
        zjp4neu    = c*z(j+4);
        bulgex     = s*z(j+4);
        z(j+4)     = zjp4neu;
    end
    S(1:sn,[j+2 j+3])     = S(1:sn,[j+2 j+3])*[c -s; s c];
    S(1:sn,[n+j+2 n+j+3]) = S(1:sn,[n+j+2 n+j+3])*[c -s; s c];
end
if j < n-1
    %
    % Givens elimination type II to delete bulge in (j+2,n+j),
    %
    [c,s]      = givens(z(j+1),bulge7);
    c2         = c*c;
    s2         = s*s;
    cs         = c*s;
    vjp1neu    = c2*v(j+1) + 2*cs*bulge4 + s2*v(j+2);
    vjp2neu    = c2*v(j+2) - 2*cs*bulge4 + s2*v(j+1);
    bjp1neu    = c2*b(j+1) + 2*cs*z(j+2) + s2*b(j+2);
    bjp2neu    = c2*b(j+2) - 2*cs*z(j+2) + s2*b(j+1);
    zjp1neu    = c*z(j+1) + s*bulge7;
    zjp2neu    = (c2-s2)*z(j+2) + cs*(b(j+2) -b(j+1));
    bulge2neu  = c*bulge2 + s*bulge3;
    bulge3neu  = c*bulge3 - s*bulge2;
    bulge4neu  = (c2-s2)*bulge4 +cs*(v(j+2)-v(j+1));
    v(j+1)     = vjp1neu;
    v(j+2)     = vjp2neu;
    b(j+1)     = bjp1neu;
    b(j+2)     = bjp2neu;
    z(j+1)     = zjp1neu;
    z(j+2)     = zjp2neu;
    bulge2     = bulge2neu;
    bulge3     = bulge3neu;
    bulge4     = bulge4neu;
    if j < n-2
        djp1neu    = c2*d(j+1) + cs*(bulge1+bulge10) + s2*d(j+2);
        djp2neu    = c2*d(j+2) - cs*(bulge1+bulge10) + s2*d(j+1);
        zjp3neu    = c*z(j+3) - s*bulge9;
        bulge1neu  = c2*bulge1 + cs*(d(j+2)-d(j+1)) - s2*bulge10;
        bulge5neu  = c*bulge5 + s*bulge6;
        bulge6neu  = c*bulge6 - s*bulge5;
        bulge7neu  = c*bulge9 + s*z(j+3);
        bulge10neu = c2*bulge10 + cs*(d(j+2)-d(j+1)) - s2*bulge1;
        bulge11neu = c*bulge11 + s*bulge12;
        bulge12neu = c*bulge12 - s*bulge11;
        d(j+1)     = djp1neu;
        d(j+2)     = djp2neu;
        z(j+3)     = zjp3neu;
        bulge1     = bulge1neu;
        bulge5     = bulge5neu;
```

```
                bulge6     = bulge6neu;
                bulge7     = bulge7neu;
                bulge10    = bulge10neu;
                bulge11    = bulge11neu;
                bulge12    = bulge12neu;
            else  % j = n-2
                djp1neu    = c2*d(j+1) + cs*(bulge1+bulge12) + s2*d(j+2);
                djp2neu    = c2*d(j+2) - cs*(bulge1+bulge12) + s2*d(j+1);
                bulge1neu  = c2*bulge1 + cs*(d(j+2)-d(j+1)) - s2*bulge12;
                bulge10neu = c2*bulge12 + cs*(d(j+2)-d(j+1)) - s2*bulge1;
                d(j+1)     = djp1neu;
                d(j+2)     = djp2neu;
                bulge1     = bulge1neu;
                bulge10    = bulge10neu;
            end
            if j < n-3
                bulge8neu  = s*bulgex;
                bulge9neu  = c*bulgex;
                bulge8     = bulge8neu;
                bulge9     = bulge9neu;
            end
            S(1:sn,[j+1 j+2])     = S(1:sn,[j+1 j+2])*[c -s; s c];
            S(1:sn,[n+j+1 n+j+2]) = S(1:sn,[n+j+1 n+j+2])*[c -s; s c];
        end
    else
        error = 1;
    end
    j = j + 1;
end
```