Computer Physics Communications 221 (2017) 304–316

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

# Diagonalization of complex symmetric matrices: Generalized Householder reflections, iterative deflation and implicit shifts☆

J.H. Noble [a], M. Lubasch [b], J. Stevens [a], U.D. Jentschura [a,*]

[a] Department of Physics, Missouri University of Science and Technology, Rolla, MO 65409, USA
[b] Clarendon Laboratory, University of Oxford, Parks Road, Oxford OX1 3PU, United Kingdom

ARTICLE INFO

*Article history:*
Received 7 October 2013
Received in revised form 9 June 2017
Accepted 17 June 2017
Available online 5 August 2017

*Keywords:*
Complex symmetric matrix diagonalization
Indefinite inner product
Implicit shift
Deflation techniques

ABSTRACT

We describe a matrix diagonalization algorithm for complex symmetric (not Hermitian) matrices, $\underline{A} = \underline{A}^{\mathrm{T}}$, which is based on a two-step algorithm involving generalized Householder reflections based on the indefinite inner product $\langle \underline{u}, \underline{v} \rangle_* = \sum_i u_i v_i$. This inner product is linear in both arguments and avoids complex conjugation. The complex symmetric input matrix is transformed to tridiagonal form using generalized Householder transformations (first step). An iterative, generalized QL decomposition of the tridiagonal matrix employing an implicit shift converges toward diagonal form (second step). The QL algorithm employs iterative deflation techniques when a machine-precision zero is encountered "prematurely" on the super-/sub-diagonal. The algorithm allows for a reliable and computationally efficient computation of resonance and antiresonance energies which emerge from complex-scaled Hamiltonians, and for the numerical determination of the real energy eigenvalues of pseudo-Hermitian and $\mathcal{PT}$-symmetric Hamilton matrices. Numerical reference values are provided.

**Program summary**
*Program Title:* HTDQLS
*Program Files doi:* `http://dx.doi.org/10.17632/x24wjxtrsg.1`
*Licensing provisions:* GPLv3
*Programming language:* Fortran 90 using fixed form notation
*Nature of problem:* Calculating the eigenvalues and optionally the eigenvectors of complex symmetric (non-Hermitian), densely populated matrices.
*Solution method:* The complex symmetric (not Hermitian) input matrix is diagonalized in two steps. First step: The matrix is tridiagonalized via a series of $(n-2)$ generalized Householder reflections, where $n$ is the rank of the input matrix. Second step: The tridiagonal matrix is diagonalized via a generalization of the "chasing the bulge" technique, which is an iterative process utilizing an implicitly shifted initial rotation followed by $(n-2)$ Givens rotations. This technique is an implementation of QL factorization, and converges roughly as $[(\lambda_i - \sigma_i)/(\lambda_{i+1} - \sigma_i)]^j$ where $\lambda_i$ is the eigenvalue located in the $(i, i)$ position of the final diagonal matrix and the eigenvalues are ordered $(|\lambda_1| < |\lambda_2| < \ldots < |\lambda_n|)$, and $j$ is the iteration. The "educated guess" $\sigma_i$ for the eigenvalue $\lambda_i$ is obtained from the analytic determination of the eigenvalues of $(k \times k)$-submatrices of $\underline{A}$, in the vicinity of the $i$th element, where $k = 0, 1, 2, 3$ (here, $k = 0$ means that the implicit shift vanishes, $\sigma_i = 0$). The routine optionally calculates the rotation matrix $\underline{Z}$, such that $\underline{Z}^{-1}\,\underline{A}\,\underline{Z} = \underline{D}$ where $\underline{A}$ is the input matrix and $\underline{D}$ is the diagonal matrix containing the eigenvalues. The $i$th column of $\underline{Z}$ then is the eigenvector of $\underline{A}$ corresponding to the eigenvalue found at the element $\underline{D}(i, i)$, in the $i$th position on the diagonal of the matrix $\underline{D}$.
*Unusual features:*
For simplicity, the "wrapper" program which contains an example application and the HTDQLS routine are distributed in the same file.



## 1. Introduction

The development of algorithms for the diagonalization of matrices is the subject of a number of standard monographs [1–3]. The degree of specialization of an algorithm depends on two

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author.
*E-mail address:* ulj@mst.edu (U.D. Jentschura).

http://dx.doi.org/10.1016/j.cpc.2017.06.014
0010-4655/

factors: **(i)** the properties of the input matrix, and **(ii)** the subset of eigenvalues and eigenvectors one is interested in. E.g., the Jacobi method [4] is based on iterative rotations affecting off-diagonal elements of a matrix ("Jacobi rotations"), with the goal of zeroing (eventually all) off-diagonal elements to machine precision after a sufficient number of iterations. The QL and QR algorithms iterate similarity transformations of a matrix [5–7] based on decompositions of the input matrix into orthogonal (Q) and upper triangular (R) or lower triangular (L) matrices. Variants of this method concern the original LR decomposition found by Rutishauser [8], and the iterative Cholesky decomposition; all of these methods (and generalizations therefore) have been summarized in Refs. [9,10] under the name of GR decompositions (for general G). When the matrix is diagonalized to machine precision, a fixed point of the similarity transformation is reached, and the algorithm stops. If the input matrix is tridiagonal, one can show [1,11] that the rate of convergence in each iteration goes as $\lambda_i/\lambda_{i+1}$, for an ordered sequence of eigenvalues $|\lambda_1| < |\lambda_2| < \cdots < |\lambda_n|$ of an $(n \times n)$-input matrix. Other methods include divide-and-conquer strategies (Sec. 8.5.4 of Ref. [2]) and bisection algorithms (Sec. 8.5.1 of Ref. [2]). The Lanczos method (Sec. 9.3 of Ref. [2]) uses so-called Krylov subspaces, which are tailored to represent approximations to eigenvectors of large eigenvalues. The algorithm typically finds the first 10% of the eigenvalues of largest modulus of a matrix with ease. The power method (Sec. 7.3.1 of Ref. [2]) is designed to converge to the largest eigenvalue of a matrix.

Here, we are concerned with algorithms for the complex symmetric eigenproblem, remembering that any square complex symmetric matrix is similar to a complex symmetric matrix, as pointed out by Gantmakher (see Corollary I. in Chap. 11.3 Ref. [12]). Let us assume that it is possible to transform a complex symmetric input matrix $\underline{A}$ using similarity transformations into the form $\underline{A} = \underline{Q}\,\underline{D}\,\underline{Q}^{-1}$, where $\underline{D}$ is a diagonal matrix and $\underline{Q}$ is orthogonal, i.e., $\underline{Q}^{\mathrm{T}} = \underline{Q}^{-1}$ (this is at variance with Hermitian matrices where we would otherwise have $\underline{Q}^{+} = \left(\underline{Q}^{\mathrm{T}}\right)^{*} = \underline{Q}^{-1}$). In view of the relation $\underline{A}\,\underline{Q} = \underline{Q}\,\underline{D} = (\underline{D}\,\underline{Q}^{\mathrm{T}})^{\mathrm{T}}$, the column vectors of the matrix $\underline{Q}$ contain the eigenvectors of $\underline{A}$, whereas the diagonal elements of $\underline{D}$ are the eigenvalues of $\underline{A}$. Furthermore, the eigenvector in the $i$ th column of $\underline{Q}$ corresponds to the eigenvalue in the $i$ th column or row of $\underline{D}$. The matrix $\underline{Q}$ which diagonalizes the input matrix $\underline{A}$ (typically, to machine precision) is constructed iteratively. In its most basic version, the QL algorithm [2,3] implements the similarity transformations by first calculating the decomposition of a real input matrix $\underline{A}$, as given by $\underline{A} = \underline{Q}\,\underline{L}$ where $\underline{Q}$ is an orthogonal matrix ($\underline{Q}^{\mathrm{T}} = \underline{Q}^{-1}$), and $\underline{L}$ is a lower left triangular matrix. One then implements the similarity transformations by simply calculating $\underline{A}' = \underline{L}\,\underline{Q} = \underline{Q}^{-1}\,\underline{A}\,\underline{Q}$ in the next step of the iteration. This corresponds to an iterative similarity transformation $\underline{A} = \underline{Q}\,\underline{A}'\,\underline{Q}^{-1} = \underline{Q}\,\underline{Q}'\,\underline{A}''\,\underline{Q}'^{-1}\underline{Q}^{-1}$ and so on. Finally, $\underline{A}$ converges to a diagonal matrix. QL factorization is known to be a rather efficient algorithm for wide classes of input matrices [2,3].

Indeed, decompositions of the above mentioned types will be used in our algorithm which is based on a combination of generalized Householder transformations (first step) together with Givens rotations (second step) in order to diagonalize an input matrix. Both the generalized Householder transformations as well as the Givens rotations describe similarity transformations involving orthogonal "rotation" matrices. In the calculation of the eigenvalues and eigenvectors of the input matrix $\underline{A}$, we proceed as follows. **(i)** The matrix $\underline{A}$ is iteratively tridiagonalized by a series of generalized Householder reflections based on the indefinite inner product ($c$-product). The indefinite inner product avoids complex conjugation in both arguments. This first step of the algorithm can be paradigmatically summarized in the equation $\underline{T} = \underline{Z}^{-1}\,\underline{A}\,\underline{Z}$, where $\underline{Z} = \underline{H}_{n-1}\,\underline{H}_{n-2}\,\ldots\,\underline{H}_2$ is a product of $n-2$ generalized Householder reflections ($\underline{Z}$ is orthogonal, with $\underline{Z}^{\mathrm{T}} = \underline{Z}^{-1}$). Also, $\underline{T}$ is a tridiagonal matrix, while $\underline{A}$ is the input matrix. In the generalized transformations, one replaces the canonical inner product $\langle \underline{u}, \underline{v} \rangle = \sum_i u_i^{*}\, v_i$ by the expression $\langle \underline{u}, \underline{v} \rangle_{*} = \sum_i u_i\, v_i$. **(ii)** After obtaining $\underline{T}$, there are a number of algorithms available in order to carry out the remaining diagonalization step. One possibility consists in an iterative "chasing the bulge" strategy (see Sec. 8.13 of Ref. [3]), using an implicit shift with Givens rotations in order to calculate the eigenvalues of the tridiagonal matrix $\underline{T}$. Roughly, one calculates a guess $\sigma$ for a specific eigenvalue $\lambda$ of $\underline{T}$. One then calculates the QL decomposition of the tridiagonal matrix $\underline{T}$ minus the guess $\sigma$ for the eigenvalue, $\underline{T} - \sigma\,\mathbb{1}_{n\times n} = \underline{Q}\;\underline{L}$. Here, $\underline{Q}$ is calculated as a product of Givens rotations. The iteration proceeds by calculating $\underline{T}' = \underline{L}\,\underline{Q} + \sigma\,\mathbb{1}_{n\times n} = \underline{Q}^{-1}\;\underline{T}\;\underline{Q}$. This procedure eventually leads to a diagonalization of $\underline{T}$ (up to machine precision), while preserving the tridiagonal structure in every iteration; this is achieved by restoring the tridiagonal form after a "bulge" introduced by an initial rotation is "chased" and annihilated by a series of Givens rotations (see Section 3.2). If an element of the super- or sub-diagonal of $\underline{T}$ other than the "targeted" eigenvalue $\lambda \approx \sigma$ is accidentally zeroed to machine precision, during an intermediate iteration of the QL decomposition, then the technique of "deflation" is used in order to subdivide the input matrix into two smaller matrices, each of which is diagonalized separately. For large matrices with an irregular structure, it is sometimes necessary to apply the deflation procedure recursively (see Section 3.3).

In our implementation, we strive for a good balance of efficiency and transparency of the `FORTRAN` code. We do not imply that the implemented algorithm is necessarily the optimal solution for any computational problem involving complex symmetric matrices. For example, if only a specific eigenvalue, say the ground state energy, of a sparsely populated matrix is desired, then "shooting" techniques, such as the Arnoldi method or its variants [13,14] can be employed. By contrast, in many applications, it is necessary to find the entire pseudo-spectrum; a particularly important example is the Green's function of a quantum system, expressed in terms of resonance and anti-resonance eigenvalues [15,16]. According to common wisdom, the spectrum of simple atoms like hydrogen or helium consists of (an infinite number of) discrete "bound-state" eigenvalues with $E < 0$ and a continuum of "free" states with $E > 0$. Canonically, the zero of the energy is normalized to a free particle at rest, with momentum eigenvalue zero [17,18]. When the Hamiltonian of the system is represented numerically by a (necessarily finite-dimensional) matrix, obtained by the projection of a basis set of quantum mechanical trial functions onto the Hamiltonian operator (see Section 4.2 of this article) or by a lattice representation [19], the "pseudo-spectrum" typically yields excellent approximations for the energy of the ground state, and of low-lying states, whereas very highly excited ("Rydberg") states and the continuum are approximated by a series of discrete energy eigenvalues and corresponding eigenvectors. Nevertheless, the entire "pseudo-spectrum" can be used for the calculation of one-loop and two-loop Bethe logarithms [20,21], and other physical processes in helium [22]. In these calculations, the "pseudo-spectrum" approximates the intermediate, "virtual" states of the atom. Our algorithm is particularly effective for the calculation of the entire spectrum of eigenvalues of densely populated matrices, which includes cases where an entire "pseudo-spectrum" is needed for physically motivated reasons.

We describe an algorithm which we refer to as HTDQLS, which stands for "Householder-based tri diagonalization followed by (generalized) QL decomposition for complex symmetric matrices". The general utility of algorithms of this type has been envisaged before [23–26]. In Refs. [23,25], generalized Householder reflections for complex symmetric matrices have been discussed, with the aim of tridiagonalizing a complex symmetric input matrix. In

the second step, the authors suggest to diagonalize the tridiagonal matrix using an efficient algorithm, e.g., QL and QR decompositions, or the Lanczos method. In order to save computer memory, the algorithm described in Ref. [23] is implemented in such a way as to treat the real and imaginary parts of the input matrix separately. The authors of Ref. [23] mention a limiting factor due to an upper limit of 400 MB main storage, which is not relevant for currently available computer systems. Full complex arithmetic rather than arithmetic involving separate real and imaginary parts, as used in Ref. [23], is standard on modern computer systems. Hence, the general concept of the paper [23] is outdated in view of technological advances. Generalized Householder reflections are proposed in Eq. (1) of Ref. [23]. We should mention that the expression $w\, w^t$ in the Eq. (1) of Ref. [23] should be understood as the dyadic product $\underline{w} \otimes \underline{w}^{\mathrm{T}}$ (in our notation), and the normalization denominator in the definition of $\underline{w}$ can be omitted, because $w = v/\|v\|_2$ already is normalized (in the notation of Ref. [23]). In Refs. [24,26], algorithmic implementation details are left out from the discussion. Here, in addition to discussing a practically useful implementation, we attempt to augment the discussion by iterative deflation techniques which take care of "premature" convergence of certain off-diagonal entries in the second step of the algorithm, and we study variants of the implicit shift $\sigma$ which is used to accelerate the convergence of the QL factorizations toward the eigenvalues. A multi-precision implementation and possibilities for a parallelization of the algorithm are also discussed (see Appendices B and C).

The organization of this paper is as follows. In Section 2, we briefly review generalized Householder reflections. We then proceed to generalized Householder tridiagonalization (HTD), which is followed by the QL decomposition with implicit shift (QLS) in Section 3. An overview of a FORTRAN implementation of the algorithm, as well as numerical reference values, are provided in Section 4. Finally, conclusions are drawn in Section 5. The appendices are devoted to the lack of positive-definiteness of the indefinite inner product, and to multi-precision implementations and parallelization.

## 2. Generalized Householder reflections

For complex symmetric matrices (these are not Hermitian), one cannot use the usual definition of the Householder reflection, which is designed for (potentially complex) Hermitian matrices. In order to apply the Householder reflections to a complex symmetric matrix, we turn to the indefinite inner product, which avoids complex conjugation,

$$\langle \underline{x}, \underline{y} \rangle_* = \underline{x}^{\mathrm{T}} \cdot \underline{y}\,, \qquad \langle \underline{x}, \underline{A}\,\underline{y} \rangle_* = \langle \underline{A}^{\mathrm{T}}\,\underline{x}, \underline{y} \rangle_*\,, \tag{1}$$

where $\underline{A}^{\mathrm{T}}$ is the transpose of $\underline{A}$. A Householder matrix employing the indefinite inner product (1) is

$$\underline{H}_{\underline{v}} = \mathbb{1}_{n\times n} - \frac{2}{\langle \underline{v}, \underline{v} \rangle_*}\, \underline{v} \otimes \underline{v}^{\mathrm{T}}\,, \qquad \underline{H}_{\underline{v}}\, \underline{x} = \underline{x} - 2\, \underline{v}\, \frac{\langle \underline{v}, \underline{x} \rangle_*}{\langle \underline{v}, \underline{v} \rangle_*}\,, \qquad |\underline{v}|_* = \sqrt{\langle \underline{v}, \underline{v} \rangle_*}, \tag{2}$$

where $\otimes$ is the tensor (dyadic) product. The transpose is indicated for the second vector for absolute clarity and in order to distinguish the formalism from the one used for Hermitian matrices. The branch cut of the square root function is chosen to be along the negative real axis. This definition is in agreement with the first (unnumbered) equation in [25], as well as Eq. (1) of Ref. [23]. The generalized Householder reflection matrices are symmetric, $\underline{H}_{\underline{v}} = \underline{H}_{\underline{v}}^{\mathrm{T}}$. Furthermore, while complex, the generalized Householder matrices are orthogonal (as opposed to unitary),

$$\underline{H}_{\underline{v}}^2 = \mathbb{1}_{n\times n} - \frac{4}{|\underline{v}|_*^2}\, \underline{v} \otimes \underline{v}^{\mathrm{T}} + \frac{4}{|\underline{v}|_*^4} \underline{v} \otimes \left( |\underline{v}|_*^2\, \underline{v}^{\mathrm{T}} \right) = \mathbb{1}_{n\times n}. \tag{3}$$

As in Ref. [23,25], the vector $\underline{v} = \underline{y} \pm |\underline{y}|_*\, \hat{e}_n$ is chosen such that $|\underline{v}|_*$ is maximal, thus

$$\underline{v} = \underline{y} \pm |\underline{y}|_*\, \hat{e}_n\,, \qquad \underline{H}_{\underline{v}}\, \underline{y} = \mp |\underline{y}|_*\, \hat{e}_n. \tag{4}$$

The projection property is easily verified,

$$\begin{aligned} \underline{H}_{\underline{v}}\, \underline{y} &= \underline{y} - \frac{2}{|v|^2}\, \left( \underline{v}^{\mathrm{T}} \cdot \underline{y} \right)\, \underline{v} \\ &= \underline{y} - \frac{2\, \underline{v}}{(\underline{y} \pm |\underline{y}|_*\, \hat{e}_n)^{\mathrm{T}} \cdot (\underline{y} \pm |\underline{y}|_*\, \hat{e}_n)} \left[ \left( \underline{y} \pm |\underline{y}|_*\, \hat{e}_n \right)^{\mathrm{T}} \cdot \underline{y} \right] \\ &= \underline{y} - \underline{v} = \mp |\underline{y}|_*\, \hat{e}_n. \end{aligned} \tag{5}$$

Note that the generalized modulus $|\underline{y}|_*$ is a complex number. The generalized Householder reflection projects onto the $n$ th unit direction of the argument vector $\underline{y}$.

## 3. Description of the HTDQLS algorithm

### 3.1. Householder reflections and tridiagonalization

In principle, it is possible to use a variety of methods to bring complex symmetric matrices into tridiagonal form. For instance, Cullum and Willoughby have shown that it is possible to use the Lanczos method to tridiagonalize complex symmetric matrices [27,28], yet we have chosen to employ generalized Householder reflections to accomplish the same goal. In Refs. [27,28], the elegant Lanczos methods have been described and analyzed in detail. These methods are primarily useful when a subset of eigenvalues (e.g., those of largest magnitude) are to be determined. Here, we are specifically concerned with the full tridiagonalization of the input matrix, and choose a generalization of the method of Householder transformations.

While the concept of using the generalized Householder reflections to tridiagonalize a complex symmetric matrix has been mentioned in Refs. [23,25], the implementation of the precise calculational procedure is not always made clear. In Ref. [23], because of a lack of true complex arithmetic, the complex symmetric matrix is separated into real and imaginary parts, causing each step to require two Householder reflections, as well as an additional unitary transform. By contrast, we here use an algorithm with a single generalized Householder reflection in each step. In the following, we endeavor to clarify the procedure utilized by our algorithm.

The tridiagonalization of an $(n \times n)$-complex symmetric input matrix $\underline{A}$ within the HTDQLS algorithm is accomplished by implementing $(n-2)$ generalized Householder reflections. In the beginning, one chooses $\underline{y}_{n-1}$ to be the column vector containing the first $(n-1)$ elements of the last row of the input matrix $\underline{A}$,

$$\underline{A} = \left( \begin{array}{c|c} \underline{B}_{n-1} & \underline{y}_{n-1} \\ \hline \underline{y}_{n-1}^{\mathrm{T}} & A_{n,n} \end{array} \right), \qquad \underline{y}_{n-1} = \begin{pmatrix} A_{1,n} \\ A_{2,n} \\ \vdots \\ A_{n-1,n} \end{pmatrix}. \tag{6}$$

We then set according to Eq. (2),

$$\begin{aligned} &\underline{v}_{n-1} = \underline{y}_{n-1} \pm |\underline{y}_{n-1}|_*\, \hat{e}_{n-1}\,, \\ &\underline{H}_{\underline{v}_{n-1}} = \mathbb{1}_{n\times n} - \frac{2}{\langle \underline{v}_{n-1}, \underline{v}_{n-1} \rangle_*}\, \underline{v}_{n-1} \otimes \underline{v}_{n-1}^{\mathrm{T}}. \end{aligned} \tag{7}$$

Possible complications due to non-positive-definiteness of the norm $|\underline{v}_{n-1}|_* = \sqrt{\langle \underline{v}_{n-1}, \underline{v}_{n-1} \rangle_*}$ are discussed in Appendix A. The

$(n \times n)$-matrix $\underline{H}_{n-1}$, as well as the first similarity transform are then defined as

$$\underline{H}_{n-1} = \left(\begin{array}{ccc|c} & H_{\underline{v}_{n-1}} & & \begin{matrix}0\\ \vdots \\ 0\end{matrix} \\ \hline 0 & \dots & 0 & 1 \end{array}\right), \tag{8a}$$

$$\begin{aligned} \underline{A}' &= \underline{H}_{n-1}^{-1}\,\underline{A}\,\underline{H}_{n-1} = \underline{H}_{n-1}\,\underline{A}\,\underline{H}_{n-1} \\ &= \left(\begin{array}{ccc|c} & & & 0 \\ & \underline{B}' = \underline{H}_{\underline{v}_{n-1}}\,\underline{B}\,\underline{H}_{\underline{v}_{n-1}} & & \vdots \\ & & & 0 \\ & & & \mp|\underline{y}_{n-1}|_* \\ \hline 0 & \dots & 0 \quad \mp|\underline{y}_{n-1}|_* & A_{n,n}\,, \end{array}\right), \end{aligned} \tag{8b}$$

where $\underline{A}'$ is tridiagonal in the last column and row. In practice, there is no need to calculate the whole matrix product $\underline{A}' = \underline{H}\,\underline{A}\,\underline{H}$. Instead consider the following expression

$$\begin{aligned} \underline{B}' &= \underline{H}_{\underline{v}}\,\underline{B}\,\underline{H}_{\underline{v}} = \underline{B} - \underline{v}\otimes\underline{u}^{\mathrm{T}} - \underline{u}\otimes\underline{v}^{\mathrm{T}} + 2q\,\underline{v}\otimes\underline{v}^{\mathrm{T}} \\ &= \underline{B} - \underline{v}\otimes\underline{w}^{\mathrm{T}} - \underline{w}\otimes\underline{v}^{\mathrm{T}}, \end{aligned} \tag{9}$$

where we introduce the notation

$$p = \frac{1}{2}\,|\underline{v}|_*^2\,, \qquad \underline{u} = \frac{\underline{B}\,\underline{v}}{p}\,, \qquad q = \frac{\underline{v}^{\mathrm{T}}\cdot\underline{u}}{2p}\,, \qquad \underline{w} = \underline{u} - q\underline{v}. \tag{10}$$

In order to calculate the transformed matrix $\underline{A}'$, one calculates the Householder vector $\underline{v}$, then $p$, $\underline{u}$, $q$, $\underline{w}$, $\underline{B}'$ and finally (trivially) $\underline{A}'$. Once these calculations are complete, and $\underline{A}'$ has been found, the matrix is deflated to $\underline{B}'$ which will be an $(n-1)\times(n-1)$ matrix. After a total of $(n-2)$ similarity transforms, the input matrix $\underline{A}$ becomes tridiagonal,

$$\begin{aligned} &\underline{T} = \underline{Z}^{-1}\,\underline{A}\,\underline{Z}\,, \qquad \underline{Z} = \underline{H}_{n-1}\,\underline{H}_{n-2}\,\dots\,\underline{H}_2\,, \\ &\underline{Z}^{-1} = \underline{H}_2\,\underline{H}_3\,\dots\,\underline{H}_{n-1}. \end{aligned} \tag{11}$$

### 3.2. Recursive algorithm and generalized Givens rotations

Cullum and Willoughby have treated the QL decomposition in Refs. [27,29], as well as provided an algorithm in Ref. [28]. Here we attempt to provide a more illustrative discussion. After the tridiagonalization, the final diagonalization is accomplished via a combination of implicitly shifted initial rotations with a series of generalized Givens rotations. The combination leads to a computationally efficient implementation of iterated QL decompositions. For each super-/sub-diagonal element, progressing from the top left corner down to the bottom right corner, we iterate the transformation

$$\underline{T}^{(k,i)} - \sigma_i\,\mathbb{1}_{n\times n} = \underline{Q}^{(k,i)}\,\underline{L}^{(k,i)}\,, \tag{12a}$$

$$\underline{T}^{(k+1,i)} = \underline{L}^{(k,i)}\,\underline{Q}^{(k,i)} + \sigma_i\,\mathbb{1}_{n\times n} = \underline{Q}^{(k,i)^{\mathrm{T}}}\,\underline{T}^{(k,i)}\,\underline{Q}^{(k,i)}\,, \tag{12b}$$

until the super- and sub-diagonal elements $T_{i,(i+1)}$ and $T_{(i+1),i}$ are zeroed to machine accuracy. Here, $i = 1,\dots,n-1$ covers the eigenvalues to be found (the $n$ th eigenvalue is obtained "for free" in view of the tridiagonal character of $\underline{T}$), and $k$ denotes the order of the iteration. The shift $\sigma_i$ constitutes a "guess" for the eigenvalue $\lambda_i$, and is optimized in order to enhance the rate of convergence of the algorithm. The idea is to iterate the transformation in higher orders of $k$, until convergence of the $i$ th eigenvalue $\lambda_I$ of $\underline{T}$ to machine accuracy is reached. After that, $i$ is advanced, $i \to i+1$. The algorithm starts at $i = k = 1$. The iterated QL decomposition is equivalent to a set of generalized Givens rotations [2,3], which can be implemented with the help of a "chasing the bulge" technique, to be described below. The rotations first create and then chase a "bulge" (non-vanishing element on the second super-diagonal) from the lower right corner to the upper left corner, as detailed in the following.

The recursive algorithm begins with the initial tridiagonal form

$$\underline{T} \equiv \underline{T}^{(1,1)} = \begin{pmatrix} D_1 & E_1 & & & \\ E_1 & D_2 & E_2 & & \\ & E_2 & \ddots & \ddots & \\ & & \ddots & D_{n-1} & E_{n-1} \\ & & & E_{n-1} & D_n \end{pmatrix}. \tag{13}$$

For the "educated guess" $\sigma_i \approx \lambda_i$ of an eigenvalue, we implement the following approximations. The first and perhaps most trivial choice would assume that the iterations have already resulted in small absolute values of the off-diagonal terms, so that $\sigma_i = D_i$. The so-called Wilkinson shift [1,11] is derived from one of the eigenvalues of the $(2\times 2)$-submatrix covering the indices $(i, i+1)$ of $\underline{T}$. A suitable choice is as follows,

$$\begin{aligned} \underline{M}_i &= \begin{pmatrix} D_i & E_i \\ E_i & D_{i+1} \end{pmatrix}, \\ \sigma_i^{\pm} &= D_i + E_i\left(\frac{D_{i+1}-D_i}{2\,E_i} \pm \sqrt{\left(\frac{D_{i+1}-D_i}{2\,E_i}\right)^2 + 1}\right), \end{aligned} \tag{14}$$

where we choose the shift $\sigma_i$ to be closest to $D_i$, minimizing $|D_i - \sigma_i^{\pm}|$. The minimization of $|D_i - \sigma_i^{\pm}|$ is indicated because the "target eigenvalue" is $D_i$. For large matrices, we observe that an improved shift based on the exact eigenvalues of the $(3\times 3)$-submatrix $(i \le n-2)$,

$$\underline{M}_i^{(3)} = \begin{pmatrix} D_i & E_i & 0 \\ E_i & D_{i+1} & E_{i+1} \\ 0 & E_{i+1} & D_{i+2} \end{pmatrix}, \tag{15}$$

may, under certain conditions, overcompensate the additional computational cost involved in solving the cubic eigenvalue equation. By the standard formulas for the solutions of a cubic (eigenvalue) equation, the eigenvalues $\Lambda_{i,m}$ of $\underline{M}_i^{(3)}$ (with $m = 1, 2, 3$) are given as follows,

$$\begin{aligned} \Lambda_{i,1} &= \frac{1}{3}\,(D_i + D_{i+1} + D_{i+2}) + \frac{2^{1/3}}{3}\,\frac{Y}{\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}} \\ &\quad - \frac{\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}}{3\cdot 2^{1/3}}\,, \end{aligned} \tag{16a}$$

$$\begin{aligned} \Lambda_{i,2} &= \frac{1}{3}\,(D_i + D_{i+1} + D_{i+2}) - \frac{(1+\mathrm{i}\sqrt{3})\,Y}{3\cdot 2^{1/3}\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}} \\ &\quad + \frac{(1-\mathrm{i}\sqrt{3})\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}}{6\cdot 2^{1/3}}\,, \end{aligned} \tag{16b}$$

$$\begin{aligned} \Lambda_{i,3} &= \frac{1}{3}\,(D_i + D_{i+1} + D_{i+2}) - \frac{(1-\mathrm{i}\sqrt{3})\,Y}{3\cdot 2^{1/3}\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}} \\ &\quad + \frac{(1+\mathrm{i}\sqrt{3})\left[Z + \sqrt{Z^2+4Y^3}\right]^{1/3}}{6\cdot 2^{1/3}}\,, \end{aligned} \tag{16c}$$

$$\begin{aligned} Y &= -(D_i + D_{i+1} + D_{i+2})^2 + 3\,[D_{i+1}\,D_{i+2} \\ &\quad + D_i\,(D_{i+1} + D_{i+2}) - E_i^2 - E_{i+1}^2]\,, \end{aligned} \tag{16d}$$

$$\begin{aligned} Z &= -(D_i + D_{i+1} - 2D_{i+2})\,[(2D_i - D_{i+1} - D_{i+2}) \\ &\quad \times (D_i - 2D_{i+1} + D_{i+2}) + 9\,E_i^2] \\ &\quad + 9\,(2D_i - D_{i+1} - D_{i+2})\,E_{i+1}^2. \end{aligned} \tag{16e}$$

The shift $\sigma_i$ is chosen as the eigenvalue $\Lambda$ of $\underline{M}_i^{(3)}$ closest to $D_i$ (minimizing the modulus $|D_i - \Lambda|$).

The generalization of the Wilkinson shift to larger sub-matrices is also utilized in Ref. [30]. The algorithm advocated in Ref. [30] is rather based on "chasing" a $(k \times k)$-bulge than modified shift parameters [see Eq. (2.1) of Ref. [30]], but algorithm **S4** on p. 101 of Ref. [30] generalizes Wilkinson's shift. The use of the eigenvalues of the trailing $(k \times k)$-matrix is a natural extension of the multi-shift program. In extensive tests of the algorithm, we found that different shifts seem to be optimal for different classes of input matrices. In typical cases, the "cubic" shift (based on the eigenvalues of a $3 \times 3$ submatrix) seems to be better suited for ill conditioned matrices, while the Wilkinson shift (based on the eigenvalues of a $2 \times 2$ submatrix) performs better for banded input matrices, with a regular (smooth) dependence of the matrix elements on their indices. For well conditioned matrices, the difference in performance between the "cubic" and the Wilkinson shifts is negligible. Due to the convergence gained by using certain shifts based on the type of matrix to be diagonalized, the distributed FORTRAN code includes the variable SHIFTMODE, with possible values SHIFTMODE = 0 (no shift, $\sigma_i = 0$), as well as SHIFTMODE = 1 [derived from the $(1\times1)$-submatrix, $\sigma_i = D_i$ ], and also the Wilkinson (SHIFTMODE = 2) and cubic shifts (SHIFTMODE = 3). The linear shift and the option to neglect having a shift at all are included for completeness. In typical cases, we observe that the total elimination of a shift (SHIFTMODE = 0) is computationally disadvantageous.

For the first rotation of the tridiagonal input matrix $\underline{T} \equiv \underline{T}^{(1,1)}$, one starts from the element $\sigma_1$ and calculates the first rotation matrix $\underline{R}$ as follows,

$$\underline{R} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c & s \\ & & & -s & c \end{pmatrix}, \qquad c^2 + s^2 = 1\,,$$

$$\underline{R}^{\mathrm{T}}\,\underline{R} = \mathbb{1}_{n\times n}\,, \tag{17a}$$

$$c = \frac{D_n - \sigma_1}{\sqrt{(D_n - \sigma_1)^2 + E_{n-1}^2}}\,, \qquad s = \frac{E_{n-1}}{\sqrt{(D_n - \sigma_1)^2 + E_{n-1}^2}}\,. \tag{17b}$$

In the initial step of the "chasing of the bulge" (which eventually leads to the calculation of $\underline{T}' = \underline{R}^{\mathrm{T}}\,\underline{T}\,\underline{R}$), one creates an tridiagonal (super-super- and sub-sub-diagonal) elements in $\underline{T}$.

We notice that the initial rotation matrix $\underline{R}$ constitutes a Givens rotation, as defined on p. 100 of Ref. [3] in terms of the elements that it eliminates from a matrix. Essentially, a Jacobi rotation eliminates the same matrix element that was used in the construction of the rotation matrix, whereas a Givens rotation eliminates a different element. The rotation matrix $\underline{R}$ constitutes a Givens rotation in the sense that the matrix $\underline{R}^{\mathrm{T}}\,(\underline{T}^{(k,i)} - \sigma_i\,\mathbb{1}_{n\times n})$ has a zero entry on the last element of the superdiagonal; however, in the course of the algorithm discussed here, we are interested in the matrix $\underline{T}'^{(k,i)} = \underline{R}^{\mathrm{T}}\,\underline{T}^{(k,i)}\,\underline{R}$. In $\underline{T}'^{(k,i)}$, the rotation $\underline{R}$ *creates* rather than *eliminates* a super-super- and sub-sub-diagonal matrix element, temporarily converting the matrix to pentadiagonal form, creating a "bulge". Both here as well as in the FORTRAN implementation, we are using the notation from Eq. (13), i.e., $\underline{D}$ is an $n$-dimensional vector containing the diagonal elements of our matrix and $\underline{E}$ is an $(n-1)$-dimensional vector containing the sub-diagonal elements of our matrix. We are calculating the initial rotation along with the Givens rotations (which eventually describe the orthogonal matrix $Q$) on the basis of the "shifted" matrices [in the sense of Eq. (12)], avoiding the necessity to calculate the shifted matrix $\underline{T}^{(k,i)} - \sigma_i\,\mathbb{1}_{n\times n}$ itself. Indeed, the shift [2,3] may be applied to the rotation matrix only, justifying the name "implicit shift".

The first transformed matrix $\underline{T}'^{(k,i)} = \underline{R}^{\mathrm{T}}\,\underline{T}^{(k,i)}\,\underline{R}$ has the form

$$\underline{T}'^{(k,i)} = \begin{pmatrix} \ddots & \ddots & & & \\ \ddots & D'_{n-3} & E'_{n-3} & & \\ & E'_{n-3} & D'_{n-2} & E'_{n-2} & F' \\ & & E'_{n-2} & D'_{n-1} & E'_{n-1} \\ & & F' & E'_{n-1} & D'_n \end{pmatrix}, \tag{18}$$

with an obvious "bulge" (the off tridiagonal element $F' = T'^{(k,i)}_{n-2,n} = T'^{(k,i)}_{n,n-2}$ is not equal to zero). The notational identification

$$\underline{G}_{n-1} \equiv \underline{R} \tag{19}$$

is consistent with the Givens rotations that follow [see also Eq. (23) below]. In order to annihilate $T'^{(k,i)}_{n,n-2} = T'^{(k,i)}_{n-2,n}$, one now uses a Givens rotation [31]. In full consistency with the discussion on p. 100 of Ref. [3], the Givens rotation eliminates the element $T'^{(k,i)}_{n,n-2}$, while the Givens rotation matrix itself has a nonzero entry at position $(n-1, n-1)$,

$$\underline{G}_{n-2} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & c & s \\ & & & -s & c \end{pmatrix}, \qquad c^2 + s^2 = 1\,,$$

$$\underline{G}_{n-2}^{\mathrm{T}}\,\underline{G}_{n-2} = \mathbb{1}_{n\times n}\,. \tag{20a}$$

$$c = \frac{E'_{n-1}}{\sqrt{E'^2_{n-1} + T'^2_{nn-2}}}\,, \qquad s = \frac{T'_{nn-2}}{\sqrt{E'^2_{n-1} + T'^2_{nn-2}}}\,. \tag{20b}$$

Notice that $c^2 + s^2 = 1$, while for Hermitian matrices the condition would otherwise read $|c|^2 + |s|^2 = 1$. We generalize the Givens rotations to preserve the complex symmetric (not Hermitian) structure of the matrix. Applying this rotation to our matrix we eliminate the "bulge"; however, a new "bulge" has been created one element up along the off tridiagonal $T''^{(k,i)}_{n-1,n-3} = T''^{(k,i)}_{n-3,n-1} \neq 0$. A total of $(n-2)$ Givens rotations are used to eliminate the "bulge" completely and return to tridiagonal form. In each step, the matrix elements are updated as

$$\begin{aligned} D'_{i+1} &= c^2 D_{i+1} + 2\,c\,s\,E_i + s^2 D_i\,, \\ D'_i &= c^2 D_i - 2\,c\,s\,E_i + s^2 D_{i+1}\,, \\ E'_{i+1} &= \sqrt{E_{i+1}^2 + F^2}\,, \\ E'_i &= (c^2 - s^2)E_i + c\,s\,(D_i - D_{i+1})\,, \\ E'_{i-1} &= c\,E_{i-1}\,, \qquad F' = s\,E_{i-1}\,, \end{aligned} \tag{21}$$

where $F$ is the (single nonvanishing) off tridiagonal element. For the initial rotation ($i = n-1$), we note that $E_{i+1} = E_n$ is not really an element of the matrix, and thus, it can be set equal to zero in the scheme given in Eq. (21). The equations for $c$ and $s$ for the initial rotation are given in Eq. (17), while the general equations for the following Givens rotations are

$$c = \frac{E_{i+1}}{\sqrt{E_{i+1}^2 + F^2}}\,, \qquad s = \frac{F}{\sqrt{E_{i+1}^2 + F^2}}\,. \tag{22}$$

Finally, one obtains the transformation $\underline{Q} = \underline{Q}^{(k,i)}$ (first transformation for the first element to be zeroed) from Eqs. (12) and (19),

$$\begin{aligned} \underline{T}^{(k+1,i)} &= \underline{Q}^{(k,i)^{\mathrm{T}}}\,\underline{T}^{(k,i)}\,\underline{Q}^{(k,i)}\,, \\ \underline{Q} &= \underline{Q}^{(k,i)} = \underline{R}\,\underline{G}_{n-2}\,\underline{G}_{n-3}\,\cdots\,\underline{G}_1 = \prod_{j=1}^{n-1}\underline{G}_j. \end{aligned} \tag{23}$$

The transformed matrix $\underline{T}^{(k+1,i)}$ again has tridiagonal form; the bulge has been "chased" upward until it disappears. We reemphasize that a single update of $\underline{T}^{(k,i)} \to \underline{T}^{(k+1,i)}$ requires an initial rotation $\underline{R}$ and $(n-2)$ further Givens rotations. One can show [1,11] that the convergence toward the eigenvalue $\sigma \to \lambda_i = D_i$ (where the equality holds to machine accuracy in the final iteration) is governed by a factor $\chi = (\lambda_i - \sigma)/(\lambda_{i+1} - \sigma)$ with each iteration. The ordered sequence of eigenvalues reads as $|\lambda_1| < |\lambda_2| < \cdots < |\lambda_n|$. Originally, this result was derived in Refs. [1,11] for real matrices, and an ordered sequence consisting of eigenvalues $\lambda_i$. Our numerical calculations provide evidence for the expectation that a similar rate of convergence is achieved, in typical applications, for complex symmetric (not Hermitian) tridiagonal matrices. In typical cases, less than 30 iterations (similarity transformations) are required to reach machine accuracy (which we understand as FORTRAN quadruple precision with roughly 32 decimals, with data type REAL*16) for a target eigenvalue $\lambda_i \approx D_i$ (where $\lambda_i = D_i$ to machine accuracy after the algorithm has converged, i.e., formally, in $\underline{T}^{(\infty,i)}$).

### 3.3. Deflation and partitioning: Reducing the matrix size

When chasing the bulge as described in Section 3.2, one strives to calculate the eigenvalues of the tridiagonal matrix $\underline{T}$ from the upper left to the lower right, i.e., one subsequently zeros (to machine accuracy) the elements $E_i$ with $i$ running from 1 to $n-1$. In the sense of Eq. (12), one iterates in ascending transformation orders $k$ with the aim of zeroing the element $E_i$ in the matrix $\underline{T}^{(k,i)}$. Let us assume that in this process an element $E_j$, with $j > i$, accidentally becomes equal to zero within machine accuracy, before $E_i$ is zeroed. This constitutes an early, or "premature", zero which requires special treatment. Namely, if we were to continue the recursive algorithm of Section 3.2 without any changes, then the bulge, which "starts in the right low corner of the matrix $\underline{T}$ and then moves upward" would always be annihilated prior to the point where it would affect $E_i$. In order to overcome the lock-up, we divide, or "partition" the matrix $\underline{T}$ into two smaller matrices,

$$\underline{T} = \begin{pmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{pmatrix}, \tag{24}$$

where $\underline{T}_1$ and $\underline{T}_2$ are tridiagonal matrices, with columns and rows running over the indices $i = 1, \ldots, j-1$ for $\underline{T}_1$ and $i = j, \ldots, n$ for $\underline{T}_2$. We assume that $\underline{Q}_1$ and $\underline{Q}_2$ diagonalize the matrices $\underline{T}_1$ and $\underline{T}_2$,

$$\underline{Q}_1^T \underline{T}_1 \underline{Q}_1 = \underline{D}_1\,, \quad \underline{Q}_2^T \underline{T}_2 \underline{Q}_2 = \underline{D}_2, \tag{25}$$

where $\underline{Q}_1$ and $\underline{Q}_2$ are the similarity transforms and $\underline{D}_1$ and $\underline{D}_2$ are the corresponding diagonal matrices of $\underline{T}_1$ and $\underline{T}_2$, respectively. We can then almost trivially construct the orthogonal transformation

$$\begin{aligned} \underline{Q} &= \begin{pmatrix} \underline{Q}_1 & 0 \\ 0 & \underline{Q}_2 \end{pmatrix}, \\ \underline{Q}^T \underline{T} \underline{Q} &= \begin{pmatrix} \underline{Q}_1^T & 0 \\ 0 & \underline{Q}_2^T \end{pmatrix} \begin{pmatrix} \underline{T}_1 & 0 \\ 0 & \underline{T}_2 \end{pmatrix} \begin{pmatrix} \underline{Q}_1 & 0 \\ 0 & \underline{Q}_2 \end{pmatrix} \\ &= \begin{pmatrix} \underline{D}_1 & 0 \\ 0 & \underline{D}_2 \end{pmatrix}. \end{aligned} \tag{26}$$

One needs to invoke the iterated, implicitly shifted QL decomposition on both of them, individually. As such we have "deflated" the matrix $\underline{T}$ into two smaller matrices. Quite surprisingly, this problem is rather scarcely treated in the literature. It is discussed very briefly in Sec. 7.11 of Ref. [3]. There are further unpublished notes that address the issue [32–34]. The solution is referred to as "deflation" in Sec. 11.4 of Ref. [33] and near the end of Sec. 3.6.2 of Ref. [34]. In Sec. 4.7 of Ref. [32], the same procedure is called "partitioning".

## 4. Program description and numerical reference data

### 4.1. Program description

The programs distributed with the CPC program library are written in FORTRAN with COMPLEX*32 arithmetic. This corresponds to roughly 32 digits (quadruple precision) for both real and imaginary parts. This enhanced arithmetic precision has been commonly available within the freely available gfortran compiler (we use version 4.9.2, see Ref. [35]), and other recent compilers, such as the IBM XL FORTRAN compiler [36]. Quadruple precision is used for the input matrix as well as for the eigenvalues and eigenvectors. Our algorithm is written transparently, with the aim of allowing for easy generalizability. In addition to the HTDQLS program, we also distribute a reference version of HTDQRS, the only difference being that the QL factorization step in Section 3.2 is replaced by QR factorization.

In the provided example cases for the algorithms (HTDQLS and HTDQRS), a $(10 \times 10)$-generalized non-Hermitian complex symmetric generalized Hilbert matrix of the form

$$A_{i,j} = \frac{\exp[-\mathrm{i}\,\theta\,(i+j-1)]}{(i+j-1)}\,, \qquad \theta = \frac{\pi}{5}, \tag{27}$$

is diagonalized. This matrix is densely populated and is ill-conditioned even for low dimensionality of the input matrix. Specifically, the ratio of the modulus of the largest as compared to the smallest eigenvalue is of the order of $10^{13}$ for a $(10 \times 10)$-generalized complex symmetric Hilbert matrix.

The algorithm is implemented using separate subroutines for the tridiagonalization and diagonalization steps, along with several other supporting subroutines and a master subroutine. There are two versions of the routine HTDQLS, used to find the eigenvalues or the eigenvalues and eigenvectors of the input matrix. In the FORTRAN source code, these routines are denoted by either a "1" (for the eigenvalue implementation) or a "2" (for the eigenvalue and eigenvector implementation) at the end of the subroutine's name. The master subroutine directs the flow of the algorithm so that the desired option is implemented. Here we briefly describe all the subroutines.

The master subroutine HTDQLS(JOBZ, N, A, D, Z, SORTFLAG, SHIFTMODE) is used to call other subroutines, and determines the order in which they are used. If JOBZ='N', then only the eigenvalues are calculated, while if JOBZ='V' then the eigenvalues and eigenvectors are calculated. The rank of the input matrix and the input matrix itself are denoted by N and A respectively. The eigenvalues and eigenvectors are returned in D and Z respectively, where the $i$ th column of Z is the eigenvector corresponding to the $i$ th eigenvalue stored in D(i). The original matrix A is unchanged by HTDQLS and may be used in order to check, e.g., the eigenvector property of the column vectors of Z, if desired, after the completion of the algorithm. Note that this "policy" is different from the one used, e.g., in LAPACK (see Ref. [37]). The boolean variable SORTFLAG determines if eigenvalues (and corresponding eigenvectors) are sorted according to the real part of the resonance energy, and the integer variable SHIFTMODE may attain the values 0, 1, 2, 3 depending on whether the implicit shift in the diagonalization routine is called in zero-shift mode, in linear, quadratic or cubic mode [see the discussion surrounding Eqs. (14) and (15)].

The routines HTD1(N, Z, D, E) and HTD2(N, Z, D, E) implement the tridiagonalization step of the program. Each routine takes the input matrix Z of rank N and tridiagonalizes it as described in Section 3.1. The tridiagonal matrix is then stored in the vectors D and E. If HTD2 is used, then Z acts both as an input as well as an output variable: at the start of the algorithm, it contains the input matrix, while Z, upon output, contains the matrix used

to tridiagonalize the input matrix. Note that the Z variable here contains a matrix different from the matrix Z used in the output of the master subroutine HTDQLS(JOBZ, N, A, D, Z, SORTFLAG, SHIFTMODE), where it otherwise contains the eigenvectors. In typical applications, the matrix used to tridiagonalize the input matrix is of no use for further calculations. If this matrix is required in an application, then one has two possibilities; either, one can call HTD2 separately, or, in the master subroutine HTDQLS, one would "intercept" the Z matrix that results from the call to HTD2, store this variable in a different matrix, e.g., ZP (for Z-prime), and output the matrix ZP in an added output variable of a modified routine HTDQLS.

The routines QLS1(N, D, E, SHIFTMODE) and QLS2(N, D, E, Z, SHIFTMODE) diagonalize the tridiagonal input matrix stored in D and E. The calculated eigenvalues are stored in D. If and when premature zeros occur the routine detects them and performs the deflation step. Finally, QLS2 stores the similarity transforms in Z.

The calculation of the implicit shift evaluated in the routine SHIFT(N, K, V, D, E, S, SHIFTMODE) depends on the number of nonzero off-diagonal elements left in the matrix/submatrix which the algorithm is working to diagonalize. Indeed, the routine SHIFT makes an educated guess on whether the quadratic or cubic shift is appropriate; we found that the cubic shift is generally advantageous for dense matrices larger than $(5 \times 5)$ while it is always advantageous to gauge the utility of all four methods (SHIFTMODE = 0, 1, 2, 3) for a particular problem at hand. Finally, the routine SHIFT then solves for the possible values of the shift, and chooses the one whose value is closest to that of the diagonal element toward which we aim to converge. The guess for the eigenvalue is returned to the appropriate version of QLS.

Finally, the routines SORT1(N, D) and SORT2(N, D, A) sort the N complex resonance eigenvalues stored in D in ascending order of their real parts. SORT2 additionally sorts the eigenvectors to match the position of the associated eigenvalues. They are only called if the boolean variable SORTFLAG in the initial call to HTDQLS evaluates to TRUE.

### 4.2. Numerical reference data

In a typical atomic physics calculation, complex symmetric matrices, $\underline{A} = \underline{A}^{\mathrm{T}}$, arise in a number of contexts. One rather straightforward example stems from the projection of a $\mathcal{PT}$-symmetric anharmonic oscillator Hamiltonian onto an appropriate set of basis states, as well as from complex scaled Hamiltonians [26,38–41]. The eigenvalues of the former are real ($\mathcal{PT}$-symmetry) or come in complex conjugate pairs ("broken" $\mathcal{PT}$-symmetry). The latter case may be an artifact due to the finite dimension of the basis set [42,43]. In natural units ($\hbar = c = \epsilon_0 = 1$), the energies take the form $E = \mathrm{Re}(E) - \mathrm{i}\Gamma/2$, where $\Gamma$ is the decay width [40]. We have checked our numerical results against published data for $\mathcal{PT}$-symmetric oscillators as well as various resonance and anti-resonance energies of anharmonic oscillators, for which accurate semi-analytic approximations ("resurgent expansions") have been found [38–41].

As an example, we consider both the real and the imaginary cubic anharmonic oscillators,

$$H_3 = -\frac{1}{2}\,\partial_x^2 + \frac{1}{2}\,x^2 + \mathrm{i}\,G\,x^3\,, \tag{28}$$

$$h_3 = -\frac{1}{2}\,\partial_x^2 + \frac{1}{2}\,x^2 + g\,x^3\,, \quad x \to x\,\mathrm{e}^{\mathrm{i}\theta}\,, \quad \partial_x \to \partial_x\,\mathrm{e}^{-\mathrm{i}\theta}\,, \quad 0 < \theta < \frac{\pi}{5}\,, \tag{29}$$

where the momentum operator is $p = -\mathrm{i}\partial_x$ and both $G$ as well as $g$ are real coupling constants. Hamiltonians of the type (28) have been extensively studied [38–44] and provide for numerically verifiable and reproducible example cases; they have been conjectured by Bessis and Zinn-Justin (1992) to have a real spectrum. The eigenvalues $E_i^{(3)}$ of $H_3$ are functions of $G$ [with $E_i^{(3)} \equiv E_i^{(3)}(G)$], while the eigenvalues $\epsilon_i^{(3)}$ of $h_3$ are functions of the coupling $g$ [with $\epsilon_i^{(3)} \equiv \epsilon_i^{(3)}(g)$ ]. Projecting the Hamiltonians onto the first few thousand eigenstates of the Harmonic oscillator, and using a multi-precision implementation of HTDQLS, we obtain 40-figure results for the lowest two eigenvalues of the cubic Hamiltonians for $G = g = 0.8, 1.0, 1.2$ (see Table 1). Extended arithmetic precision (Bailey's MPFUN package [45–47]) is used in order to calculate the 100-decimal reference value

$$E_0^{(3)}(G = 0.8) = 0.\,74094\,89714\,82359\,67140\,99523\,87680\,56298\,96492\,18672\,78632\,20295\,06972\,65779\,86489\,95262\,29285\,78562\,62734\,77203\,42411\,, \tag{30}$$

which constitutes the real (rather than complex) eigenvalue of a manifestly complex $\mathcal{PT}$-symmetric Hamiltonian. It is determined from a matrix representation of the Hamiltonian $H_3$ of relatively modest size ( $700 \times 700$), in a nonorthogonal basis, with details to be discussed below. Along similar lines, Macfarlane implemented a procedure using the Lanczos method to calculate the eigenvalues of the anharmonic-oscillators of the form $H = \frac{1}{2}(p^2 + x^2) + \lambda\,x^{2m}$, with $m$ ranging form 2 to 6 [48]. Using the Lanczos method, Macfarlane found the energies of the ground states using basis sets of around 100 to 700 functions, reaching a final numerical accuracy of about 32 digits in typical cases. The Lanczos methods employed in Ref. [48] require a good estimate of the energy prior to the implementation of the algorithm. HTDQLS on the other hand requires no prior knowledge of the spectrum to solve for the eigenvalues. A multi-precision implementation of the Lanczos algorithm has been demonstrated in Ref. [48] to reach 80-digit precision. The 100-figure result in Eq. (30) provides for a numerical verification of the unbroken $\mathcal{PT}$-symmetry of the cubic anharmonic oscillator, and strong numerical evidence for the Bessis–Zinn-Justin conjecture (see also Ref. [43]) that its eigenvalues are indeed real rather than complex.

As a further example of a $\mathcal{PT}$-symmetric and complex rotated Hamiltonian, we consider the real and imaginary quintic perturbations, which give rise to the Hamiltonians

$$H_5 = -\frac{1}{2}\,\partial_x^2 + \frac{1}{2}\,x^2 + \mathrm{i}\,G\,x^5\,, \tag{31}$$

$$h_5 = -\frac{1}{2}\,\partial_x^2 + \frac{1}{2}\,x^2 + g\,x^5\,, \quad x \to x\,\mathrm{e}^{\mathrm{i}\theta}\,, \quad \partial_x \to \partial_x\,\mathrm{e}^{-\mathrm{i}\theta}\,, \quad 0 < \theta < \frac{\pi}{7}\,. \tag{32}$$

These Hamiltonians have been studied in Ref. [40]. The eigenvalues of $H_5$ and $h_5$ are functions of $G$ and $g$, and we denote them as $E_i^{(5)}(G)$ and $\epsilon_i^{(5)}(g)$, respectively. As with the cubic perturbations, we find the ground state energies and the first excited state energies for $G = g = 0.8, 1.0, 1.2$. These values are given in Table 2. The accuracy of the 40-figure results in Table 2 is estimated based on the apparent convergence of the numerical data as the size of the matrix is increased.

The HTDQLS algorithm is best suited to the diagonalization of highly populated matrices, and this property can be exploited as follows. In order to determine the eigenvalues of the Hamiltonians given in Eqs. (28), (29), (31), and (32), we use a non-orthogonal basis, spanned by the functions

$$\psi_m(x) = \exp(-a\,m\,x^2)\,, \qquad m = 1, \ldots, \frac{n}{2}\,,$$
$$\psi_{m'}(x) = x\,\exp(-a\,m'\,x^2)\,, \qquad m' = \frac{n}{2} + 1, \ldots, n, \tag{33}$$

**Table 1**
The ground and first excited state energies for $H_3$ and $h_3$ for a number of numerical values of $G$ and $g$.

| $G$ | $E_0^{(3)}(G)$ | $E_1^{(3)}(G)$ |
|---|---|---|
| 0.8 | 0.74094897148235967140995238768056298964 92 | 2.55909365868429583433763075639490895905 03 |
| 1.0 | 0.79734260750890618903908096079101316309 72 | 2.77352498519537971540581700001553014231 08 |
| 1.2 | 0.84909706689025801543791740828425714606 28 | 2.96727359344265206608573034670452972879 69 |
| $g$ | $\epsilon_0^{(3)}(g)$ | $\epsilon_1^{(3)}(g)$ |
| 0.8 | 0.56106620897940477511692816643142268773 84 | 1.99145669889866119488438454996532512008 99 |
| | −0.35859844691200673512575409270525983934 9 i | −1.36970573628264552784152815512606349483 48 i |
| 1.0 | 0.61288843330775462425881750198866514137 333 | 2.18041383753634877123016196354174113124 71 |
| | −0.40859266693226728315949886876716051627 09 i | −1.52620765569303251000685394696749562444 59 i |
| 1.2 | 0.65947141671929912789771913415449715602 52 | 2.34789833330708248460227182869909735311 83 |
| | −0.45015003426236504630756576824437660558 19 i | −1.65990636058492374454809052851463669516 80 i |

**Table 2**
The ground and first excited state energies for $H_5$ and $h_5$ for different values of $G$ and $g$.

| $G$ | $E_0^{(5)}(G)$ | $E_1^{(5)}(G)$ |
|---|---|---|
| 0.8 | 0.75389124621589786099337755323420847558 30 | 2.72584719948185664152979718639908174280 31 |
| 1.0 | 0.79140175777864076155860859702017488502 26 | 2.87580993584575639268780020627266878199 72 |
| 1.2 | 0.82455852408502236750428497431837653777 26 | 3.00702088732252853807330439861318505358 28 |
| $g$ | $\epsilon_0^{(5)}(g)$ | $\epsilon_1^{(5)}(g)$ |
| 0.8 | 0.67450329439700818291896353707548915544 78 | 2.45493787189842310938036792862778581446 87 |
| | −0.24497261241861497579575125097610853374 74 i | −0.98294115378930202944681413329505028737 59 i |
| 1.0 | 0.70875699952222315206082074451175355469 39 | 2.59036713329602333960788878304385306013 39 |
| | −0.26762943090641375087559443630536242086 97 i | −1.06050350702232561922594400015297159643 91 i |
| 1.2 | 0.73900997993150365467103706748425891486 60 | 2.70878087811349467420450292754718908864 08 |
| | −0.28682232235861442117427312071149714709 71 i | −1.12701898384173378366059369869202592386 00 i |

where $n$ is the (even integer) total number of basis functions and $m, m'$ serve as counters. This defines basis functions $\psi_m(x)$ with $m = 1, \ldots, n$ which have even parity for $1 \le m \le n/2$ and odd parity for $n/2 < m \ne n$. We find that the choice $a = 0.04$ is useful for the calculations leading to the resonance energies in Tables 1 and 2. The $(n \times n)$ symmetric real overlap matrix $\underline{S}$ and the complex symmetric Hamiltonian matrix $\underline{H}$ have the elements

$$S_{ij} = \int_{-\infty}^{\infty} dx\, \psi_i(x)\, \psi_j(x) = \langle \psi_i | \psi_j \rangle_*\,,$$
$$H_{ij} = \int_{-\infty}^{\infty} dx\, \psi_i(x)\, H_3\, \psi_j(x) = \langle \psi_i | H_3 | \psi_j \rangle_*, \tag{34}$$

where we use the imaginary cubic perturbation given in Eq. (28) for our example, and the inner product is denoted as in Eq. (1). On the basis of the HTDQLS algorithm, we first calculate the square root of the overlap matrix,

$$\underline{S} = \underline{Q}\,\underline{D}\,\underline{Q}^{\mathrm{T}}\,. \qquad \underline{M} = \underline{Q}\,\sqrt{\underline{D}}\,\underline{Q}^{\mathrm{T}}\,, \qquad \underline{S} = \underline{M}^2. \tag{35}$$

The square root of the diagonal matrix $\underline{D}$ is easily calculated. We now use the following ansatz for an eigenvector, expressed in the non-orthogonal basis,

$$|\psi\rangle = \sum_j c_j\, |\psi_j\rangle. \tag{36}$$

The eigenvalue problem within the basis, $\sum_j c_j\, H\, |\psi_j\rangle = E \sum_j c_j\, |\psi_j\rangle$, can then be formulated as

$$\sum_j \langle \psi_i | H | \psi_j \rangle\, c_j = \sum_j E\, \langle \psi_i | \psi_j \rangle\, c_j, \tag{37}$$

With the coefficient vector $\underline{c}$ begin composed of the $c_j$, we have

$$\underline{H}\,\underline{c} = E\,\underline{S}\,\underline{c}\,, \qquad \underline{d} = \underline{M}\,\underline{c}\,, \qquad \underline{M}^{-1}\,\underline{H}\,\underline{M}^{-1}\,\underline{d} = E\,\underline{d}, \tag{38}$$

where $E$ is the resonance energy. A diagonalization of the effective Hamiltonian matrix

$$\underline{H}_{\mathrm{eff}} = \underline{M}^{-1}\,\underline{H}\,\underline{M}^{-1} \tag{39}$$

then leads to the reference values given in Eq. (30), and in Tables 1 and 2.

## 5. Computational performance of the algorithm

### 5.1. Numerical accuracy

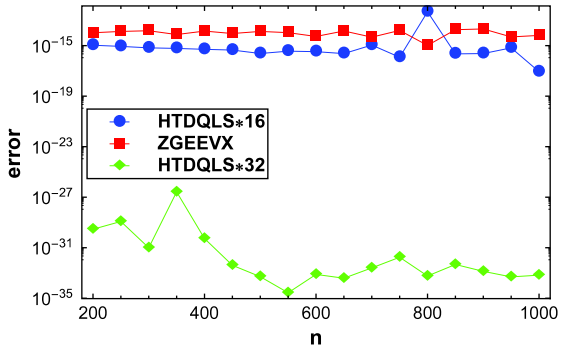
In order to gauge the numerical accuracy of the HTDQLS algorithm, we turn to a complex rotated version of the harmonic oscillator Hamiltonian,

$$H_0 = -\frac{1}{2}\,\partial_x^2 + \frac{1}{2}\,x^2\,, \qquad x \to x\,\mathrm{e}^{\mathrm{i}\theta}\,,$$
$$\partial_x \to \partial_x\,\mathrm{e}^{-\mathrm{i}\theta}\,, \qquad \theta = \frac{\pi}{16}. \tag{40}$$

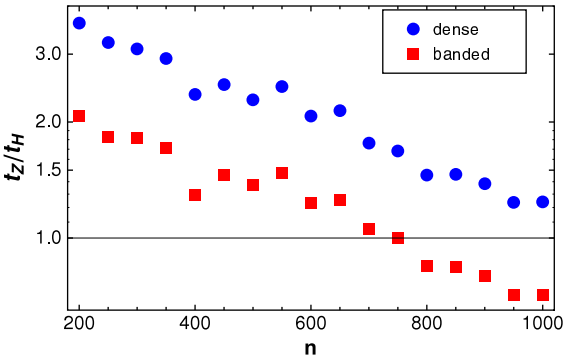
The ground-state eigenvalue of the harmonic oscillator is unaffected by the complex scaling and reads as $\lambda_0 = \frac{1}{2}$. On the other hand, using a projection of the complex rotated $H_0$ onto a suitable basis, we can generate complex symmetric matrices in which at least the first eigenvalue is known, namely, $\lambda_0$. A measure of the numerical accuracy of the method is given as follows,

$$\mathtt{err} = \frac{|D_1 - \lambda_0|}{\lambda_0}, \tag{41}$$

where err is the numerical error, and $D_1$ is the ground-state eigenvalue as found by the corresponding algorithm. The goal is to compare HTDQLS to ZGEEVX, which is a LAPACK routine [37] that diagonalizes complex matrices (in COMPLEX*16 precision, i.e., with roughly 16 significant decimals). We note that ZGEEVX does not specialize in complex *symmetric* matrices but is a more general solver. Aside from a single outlier at $n = 800$, we found that the HTDQLS algorithm is generally an order of magnitude more accurate than the LAPACK routine ZGEEVX (see Fig. 1). In typical cases, we find that the final numerical loss of our method in reproducing known eigenvalues of Hamiltonians does not exceed 4–5 decimals, consistent with the outlier in Fig. 1. For comparison, we also plot in Fig. 1 the numerical accuracy obtained using a COMPLEX*32 version of HTDQLS; such a high-precision version is not available for ZGEEVX.




(a) Numerical accuracy.




(b) Speed.

**Fig. 1.** In comparing the HTDQLS algorithm with the LAPACK routine ZGEEVX, the relative numerical accuracy of the ground state energy of the complex rotated harmonic oscillator $H_0$ given in Eq. (40) is plotted as a function of the size of the matrices [see Fig. (a)]. In Fig. (b), the average ratio of the runtimes, $t_Z/t_H$ (where $t_Z$ is the runtime of ZGEEVX and $t_H$ is the runtime of HTDQLS), is plotted against the rank of the matrices. Two types of matrices were used (densely populated and banded). Further details are in the text.

### 5.2. Speed

In order to test the computational efficiency of HTDQLS, we again compare the LAPACK routine ZGEEVX with a COMPLEX*16 version of HTDQLS. This is done with the help of two types of matrices, the first being composed of random complex numbers, leading to densely populated, complex symmetric matrices, while the second type of matrices are generated using the harmonic oscillator Hamiltonian with an imaginary cubic perturbation [see Eq. (28)], with random values of $G$, resulting in banded, complex symmetric matrices. We then average 150 trials for each rank (200 to 1000) and find the ratio of the run times (see Fig. 1(b)). For smaller matrices we found that HTDQLS runs quite a bit faster, but as the size of the matrices increases ZGEEVX's performance improves. By rank 750, ZGEEVX performs faster (albeit slightly) than HTDQLS for the banded matrices. For the densely populated matrices on the other hand, HTDQLS is faster for all the matrices we tested, but again, the performance of ZGEEVX improves with the size of the matrix.

### 5.3. Defective matrices

Defective matrices (see, e.g., p. 316 of Ref. [2]) do not have a complete basis of eigenvectors. This particular situation may occur when an eigenvalue is duplicate but one cannot find two linearly independent eigenvectors corresponding to the degenerate eigenvalue. The Hamiltonians studied here correspond to physical systems where naturally, every quantum state has to correspond to a well-defined wave function. Even if, in a physical system, an eigenvalue is degenerate, then one can argue that the full Hamiltonian of the system, acting on the infinite-dimensional Hilbert space of wave functions, cannot be defective in nature because that would be self-contradictory: Namely, it would imply the existence of a duplicate eigenvalue (two energetically degenerate states) without two corresponding eigenvectors, i.e., without two quantum states that correspond to the two "physical states". The Hamiltonians we study present finite-dimensional approximations to the full Hamiltonian operator which acts on the infinite-dimensional Hilbert space. In principle, the finite-dimensional nature of the approximation could induce defectiveness in exceptional, unfortunate cases, but we have encountered no computational problems caused by this aspect in the applications discussed here. Larger matrices will approximate the spectrum of the Hamiltonian which they model more closely, and so, one could argue that the occurrence of an accidental defectiveness in a finite-dimensional approximation to a quantum Hamiltonian becomes less and less likely the larger the dimension of the matrix is.

Nevertheless, we should explicitly state here that we exclude defective matrices from the scope of the HTDQLS routine. In order to enhance the perspective, we have performed trial runs, applying the algorithm to defective matrices. In diagonalizing defective matrices, HTDQLS typically determines the eigenvalues reliably, but the determination of the eigenvectors proves to be troublesome; typically, the algorithm finds the correct eigenvector and a nonsensical result is obtained for the non-existent one. In principle, as defective matrices either have duplicate eigenvalues, or at least one eigenvalue which is equal to zero it is simple for the algorithm to check if the input matrix was defective. In HTDQLS, in the event that a defective matrix is detected, an error message appears advising the user to verify the eigenvalues. The test is performed by verifying that the ratio of the modulus of two subsequent eigenvalues does not deviate from unity by more than EPS (epsilon), and that the absolute value of any particular eigenvalue is greater than EPS (epsilon). Note that this check is dependent on the eigenvalues having been sorted, as such if the boolean variable SORTFLAG evaluates to FALSE then the check is not performed (sorting the eigenvalues is computationally cheap). Additionally, we recall that problems associated with defective matrices relate to the eigenvectors, not eigenvalues. Hence, if the program is run in eigenvalue mode (i.e. the eigenvectors are not found) the check is not performed as it would be irrelevant.

## 6. Conclusions

In this paper, we use physically inspired generalized Householder transformations (see Refs. [23–26]) in order to construct an efficient and scalable matrix diagonalization algorithm for complex symmetric matrices. We follow the footsteps of other attempts to generalize basic matrix algorithms to complex symmetric matrices (see, e.g., Ref. [49] where the Rayleigh quotient is suitably generalized in the context of the Jacobi–Davidson algorithm). To this end, we discuss the necessary generalizations to the definition of the Householder reflection for complex symmetric matrices (Section 2).

The algorithms discussed in this paper are mainly targeted at applications in physics, and work best with densely populated matrices whose entries depend smoothly on the matrix indices. We therefore advocate a two-step approach which still allows for a relatively compact implementation. In the first step, the input matrix is tridiagonalized using $(n-2)$ generalized Householder transformations (Section 3.1 and Appendix A). In the second step, the diagonalization is achieved by shifted QL decompositions which are implemented via Givens rotations ("chasing the bulge"). The Givens rotations are generalized to orthogonal as opposed to

unitary form (cf. Refs. [2,3]). The implicit shift, which is known to lead to higher computational efficiency, is constructed using an improved variant of the Wilkinson shift [1], which uses eigenvalues determined from a cubic equation if appropriate. This is described in Section 3.2. A similarity transform is performed, creating a bulge, which is then "chased" out of the matrix. Finally, a new tridiagonal matrix is obtained. The process is then repeated until diagonalization to machine accuracy is achieved. The process is summarized in Eq. (12). Necessary partitions due to "premature zeros" along the super-/sub-diagonal of the tridiagonal matrix are described in Section 3.3.

A program description is provided in Section 4.1. Our algorithm calculates all eigenvalues of a complex symmetric matrix, as required for, e.g., calculations of the complex scaled Green function [50]. This is followed by numerical reference data for anharmonic oscillators, partially obtained from a multi-precision version of the algorithm, as given in Section 4.2. An illustration of the procedure used in obtaining accurate $\mathcal{PT}$-symmetric eigenenergies from a matrix representation of the Hamiltonian of rather modest size is given in Eqs. (33)–(39).

Finally, we compare to routines within publicly accessible libraries (e.g., LAPACK, Ref. [37]). For typical applications (matrices around rank 500), we find that our HTDQLS routine is able to compete well with LAPACK's routine ZGEEVX (see Section 5).

We have tested the algorithm on matrices with random entries (all elements are generated by a random number generator, resulting in random entries distributed over a finite interval), on anharmonic oscillators and helium (many-body atomic structure) calculations (see Section 4.2). A multi-precision implementation is discussed in Appendix B, while a discussion on the parallelization of the algorithm is relegated to Appendix C.

In summary, we would like to emphasize that the purpose of the current paper is not to compete with the highly versatile LAPACK package [37] for matrices of arbitrary structure. We rather aim to write an easily modifiable, and accessible routine that can be modified by the user for use with any multi-precision software, or as a testbed for other modifications of the algorithm that are designed to aid in the diagonalization of input matrices of a even more specific structure. For that purpose, we also include our programs in the CPC program library. For complex symmetric matrices, it seems that the potential of the indefinite inner product (1) in the construction of numerical algorithms has been somewhat underestimated, especially as it pertains to the linear algebra of complex symmetric matrices. The complex resonance energies calculated in Section 4.1 describe a quantum particle trapped in a potential in which it is possible to tunnel out to infinity in a finite amount of time, within a cubic or quintic anharmonic oscillator. Our high-precision numerical reference values obtained in Eqs. (30) and Tables 1 and 2 demonstrate the utility of the algorithm discussed here; similar calculations have been the basis for the checking of a number of conjectures regarding generalized quantization conditions describing the spectrum of anharmonic oscillators [38–41]. The ability of HTDQLS to handle densely populated matrices is sometimes useful. For more complex physical systems, such as metastable, doubly excited Rydberg states of the helium atom, modeled in a Hylleraas basis [51,52], the Hamiltonian is not sparsely populated, and the full power of a general matrix diagonalization method that is suited for densely populated matrices becomes useful.

## Acknowledgments

The authors acknowledge helpful conversations with Professors Andras Kruppa and Istvan Nandori. Also, support is acknowledged from the National Science Foundation (NSF) under grants PHY-1403973 and PHY-1710856. A Missouri Research Board Grant also is acknowledged. A precision measurement grant from the National Institute of Standards and Technology (NIST) supported early stages of this work.

## Appendix A. Indefinite inner product and zero norm

The indefinite inner product (1) is not positive definite, implying that the "length" $|\underline{y}|_*$ of a vector can be zero even if the entries of $\underline{y}$ are nonzero. Let us consider an $n \times n$ matrix

$$\underline{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 1 \\ 0 & 1 & 0 & 0 & \cdots & \mathrm{i} \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \ddots & 0 \\ 1 & \mathrm{i} & 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{42}$$

According to Eq. (6), the vector $\underline{y}_{n-1}$ which is used in the first generalized Householder transformation, reads as

$$\underline{y} = \underline{y}_{n-1} = \begin{pmatrix} 1 \\ \mathrm{i} \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad |\underline{y}|_*^2 = \underline{y}_{n-1}^{\mathrm{T}} \cdot \underline{y}_{n-1} = 0. \tag{43}$$

The "length" of the vector $\underline{y}_{n-1}$, measured in terms of the complex inner product, is zero. Furthermore, the Householder vectors $\underline{v}$, defined according to Eq. (4), have the properties

$$\underline{v} = \underline{y} \pm |\underline{y}|_* \, \hat{e}_n = \underline{y}, \qquad |\underline{v}|_* = |\underline{y}|_* = 0, \tag{44}$$

and thus zero norm, independent of the sign choice ($\pm$). The Householder matrix $\underline{H}_{\underline{v}}$ thus diverges according to

$$\underline{H}_{\underline{v}} = \mathbb{1}_{n\times n} - \frac{2}{\langle \underline{v}, \underline{v} \rangle_*} \, \underline{v} \otimes \underline{v}^{\mathrm{T}} \to \mathbb{1}_{n\times n} - \frac{2}{|\underline{y}|_*^2} \, \underline{y} \otimes \underline{y}^{\mathrm{T}} \to \infty. \tag{45}$$

The problem cannot be remedied by a permutation of the entries in the matrix (42), as an inspection easily shows. The validity of our approach thus is restricted to input matrices where the norm of the Householder vectors does not accidentally vanish during the tridiagonalization step. The regular, smooth pattern of matrix entries in physical applications, such as anharmonic oscillators and atomic structure calculations, prevents the occurrence of the Householder vectors with vanishing norm in typical cases. Our program stops with an error message if the norm vanishes.

Furthermore, cases in which the norm $|\underline{v}|_*$ is close to zero in view of mutual cancellations in the calculation of the complex inner product norm, may lead to numerical instabilities. In cases where the algorithm is applied to input matrices with an irregular pattern of matrix entries, is thus recommended to check the numerical accuracy of the eigenvectors found in the diagonalization of the matrix, by an explicit multiplication of the eigenvectors with the input matrix, at the end of the calculation, as is done in the example programs that are distributed with the FORTRAN code distributed with this article.

## Appendix B. Multi-precision

As we have already stated, one of the advantages of the HTDQLS algorithm as opposed to a similar LAPACK routine is the ease with which the algorithm can be augmented in terms of arithmetic precision. The original version of the algorithm is written in COMPLEX*32 precision, which already exceeds the accuracy utilized

in LAPACK. We can increase the precision using a multi-precision package, such as Bailey's MPFUN90 (Refs. [45–47]). According to the discussion in Section 4.2, this extension allows us to calculate eigenvalues very accurately. We find that the "legacy" version of the MPFUN package [53] offers the most straightforward possibilities for an implementation. Here we briefly discuss how the implementation is realized in the program HTDQLS_MP_EXAMPLE.f distributed with the CPC program library.

First, one should note that, when implementing the multi-precision version of the algorithm, care is required when defining constants. To that end a simple module, MPCONST defines the high precision constants used by the algorithm. The constants are defined in the obvious way, i.e., ZERO = '0.0', ONE = '1.0', TWO = '2.0', etc. The imaginary unit i is defined as II = MPCMPL (ZERO, ONE). Finally $\pi$ is defined as FOUR*ATAN(ONE), where we note that the arctangent function has been extended for use with high precision numbers.

The high-precision algorithm requires that the translation modules be used; so the statement USE MPMODULE follows all PROGRAM, SUBROUTINE, and MODULE statements in the high precision version of HTDQLS (see Ref. [47]). Immediately following this is the statement USE MPCONST for all PROGRAM and SUBROUTINE statements, including our previously defined constants. The parameters in MPMODULE and the constants in MPCONST are initiated in the main program, after all type declarations, using CALL MPINIT and CALL ICONST. In order to utilize high precision variables in HTDQLS all real and complex type declarations are replaced with MPFUN90 type declarations, i.e.

$$\texttt{REAL} * 16 \rightarrow \texttt{TYPE(MP\_REAL)},$$

$$\texttt{COMPLEX} * 32 \rightarrow \texttt{TYPE(MP\_COMPLEX)}.$$

Finally, in order to output high precision numbers, the WRITE and PRINT statements must be replaced with a call to the MPWRITE subroutine. In order to set the output precision, one calls MPSETOUTPUTPREC. In the included example (HTDQLS_MP_EXAMPLE.f) we have CALL MPSETOUTPUTPREC(57), meaning that the first 57 digits of the high precision numbers will be printed. The output style is somewhat rigid, and in the event that one wishes to adjust the style the following feature may prove useful: Namely, a multi-precision complex number X can be converted to a double precision complex number Y as follows,

$$\texttt{Y} = \texttt{DBLE(MPREAL(X))} + \texttt{DBLE(AIMAG(X))} * \texttt{II},$$

where II is the imaginary unit i.

Finally, adjusting the precision of the code is achieved by opening the file mpmod90.f and setting the parameter mpipl on line 46 equal to the desired precision. The file then needs to be recompiled (i.e. gfortran -o mpmod90.f), and then linked to the desired program that uses the multiprecision arithmetic.

## Appendix C. Parallelization

When considering the possibilities for the parallelization of HTDQLS it immediately becomes clear that priority should be given to the tridiagonalization step as it dominates the runtime (see Fig. C.1). As such, we limit this discussion of the parallelization to the tridiagonalization step HTD1. As with the Hermitian version written for ScaLAPACK [54], and the generalization to the complex symmetric case, discussed in Ref. [25], we do not worry about saving the similarity transforms. The algorithm described in Section 3.1 is clearly designed as a sequential algorithm which can be modified easily. This general setting somewhat limits our opportunities for parallelization. We are then left with two possible courses of action: We may either parallelize "inner" steps in the existing code or redesign the entire tridiagonalization algorithm with parallelization in mind. We discuss both possibilities.

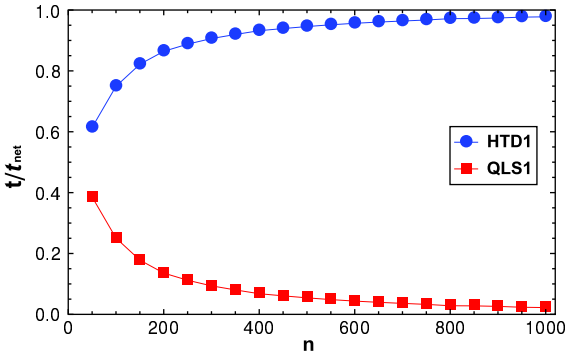



**Fig. C.1.** The fraction of the total run time ($t_{\text{net}}$) HTD1 and QLS1 is plotted as a function of the matrix size $n$. For larger $n$, the runtime is almost entirely taken up by the tridiagonalization step HTD1.

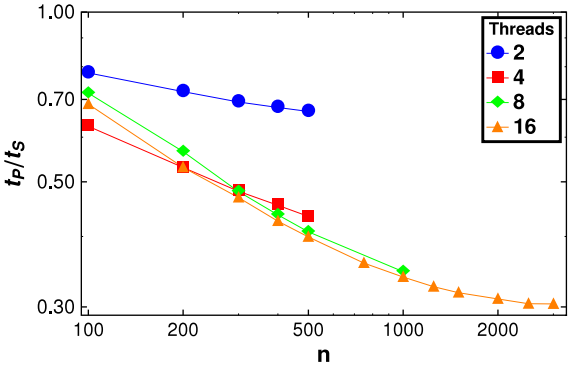



**Fig. C.2.** This plot shows the relative execution time of OpenMP HTDQLS with various thread-counts compared to the sequential version; the latter corresponds to a single OpenMP thread. The results were taken on an Intel i5 processor, which allows four threads to be executed in parallel. The relative execution time decreases as a function of $n$; the optimum value for the absolute parallel execution time over the sequential time $t_P/t_S$ is 0.25 in view of the availability of four independent cores on the processor used by us. The relative time spent in the parallelized part of the computation (the tridiagonalization) increases with $n$ [$\mathcal{O}(n^3)$ as compared to $\mathcal{O}(n^2)$]. The parallel and sequential execution times were compared for fully occupied complex symmetric matrices with random entries; ten random matrices were generated and the execution times were recorded and compared.

Let us try to write pseudocode describing HTD1 is as follows,

```
HTD1
01   for i = n, 3, -1
02      d( i ) = A( i , i )
03      y = A( i , 1:i-1 )
04      e( i-1 ) = ∓|y|_*
05      v = y ± |y|_* ê_{i-1}
06      p = (1/2)|v|_*^2
07      u = (1/p) A( 1:i-1 , 1:i-1 ) v
08      q = (1/2p) v^T·u
09      w = u - qv
10     A( 1:i-1 , 1:i-1 ) = A( 1:i-1 , 1:i-1 ) - v⊗w^T - w⊗v^T
11   d( 2 ) = A( 2 , 2 )
12   d( 1 ) = A( 1 , 1 )
13   e( 1 ) = A( 2 , 1 )
```

The operations inside the main loop depend on the previous steps, meaning that the parallelization of the largest loop is prohibited.

However, the operations on lines 07 and 10 in the above pseudocode each require doubly nested DO loops. We parallelized these operations using OpenMP [55]. The resulting parallelized algorithm was tested on a quad-core Intel i5 processor, and significant speedup was achieved (see Fig. C.2). The OpenMP implementation is limited in scope to shared memory architectures (multi-core machines) and cannot easily be ported to distributed-memory systems; the latter require the use of the MPI (message passing interface) [56].

We may draw inspiration for a possible distributed-memory implementation from the Hermitian equivalent pzhetrd which is included in ScaLAPACK [54] and pzsytr2 [25]. This approach is implemented using distributed memory MPI, and utilizes a block cyclic memory distribution (for efficient communication), where the matrix $\underline{\underline{A}}$ is distributed using column blocking, i.e. it is broken into m column blocks of size $n \times n_b$ in a stripe-like pattern, where $m=n/n_b$ (see Algorithms 15 and 16 on p. 36 of Ref. [54] and Ref. [57]). The tridiagonalization routine is converted from Level 2 BLAS (vector–vector/vector–matrix operations) to Level 3 BLAS (matrix–matrix operations), as outlined in Refs. [54] and [57]. However, ScaLAPACK does not utilize the generalized Householder reflection, which we use to tridiagonalize complex symmetric matrices, and performs the tridiagonalization beginning with the top left and moving down (the *QR* variant). We have found that tridiagonalization starting from the bottom right (the *QL* variation) is advantageous when dealing with physical systems, because the lowest eigenvalues in a discrete representation of the Hamiltonian tend to converge first. With both these considerations in mind, we include the pseudocode for the Level 3 BLAS variation of HTD1 which we refer to as HTD3;

HTD3

01 for j = m, 2, -1

02 for i = j·$n_b$, (j-1)·$n_b$+1, -1

03 $\underline{d}$( i ) = $\underline{\underline{A}}$( i , i ) - $\sum_{k=i+1}^{j\cdot n_b} 2\ \underline{v_k}(\ i\ )\ \underline{w_k}(\ i\ )$

04 $\underline{y}$ = $\underline{\underline{A}}$( 1:i-1 , i ) - $\sum_{k=i+1}^{j\cdot n_b} (\ \underline{v_k}(\ i\ )\ \underline{w_k}(\ 1:i-1\ ) + \underline{w_k}(\ i\ )\ \underline{v_k}(\ 1:i-1\ )\ )$

05 $\underline{e}$( i-1 ) = $\mp|\underline{y}|_*$

06 $\underline{v_i}$ = $\underline{y} \pm |\underline{y}|_* \hat{e}_{i-1}$

07 p = $\frac{1}{2}|\underline{v_i}|_*^2$

08 $\underline{\underline{S}}$ = $\sum_{k=i+1}^{j\cdot n_b} (\ \underline{v_k}(\ 1:i-1\ ) \otimes \underline{w_k}^T(\ 1:i-1\ ) + \underline{w_k}(\ 1:i-1\ ) \otimes \underline{v_k}^T(\ 1:i-1\ )\ )$

09 $\underline{u}$ = $\frac{1}{p}$ ( $\underline{\underline{A}}$( 1:i-1 , 1:i-1 ) - $\underline{\underline{S}}$ ) $\underline{v_i}$

10 q = $\frac{1}{2p}\ \underline{v_i}^T \cdot \underline{u}$

11 $\underline{w_i}$ = $\underline{u}$ - q$\underline{v_i}$

12 l = (j-1)·$n_b$

13 $\underline{\underline{S}}$ = $\sum_{k=1+1}^{j\cdot n_b} (\ \underline{v_k}(\ 1:l\ ) \otimes \underline{w_k}^T(\ 1:l\ ) + \underline{w_k}(\ 1:l\ ) \otimes \underline{v_k}^T(\ 1:l\ )\ )$

14 $\underline{\underline{A}}$( 1:l , 1:l ) = $\underline{\underline{A}}$( 1:l , 1:l ) - $\underline{\underline{S}}$

15 use HTD1 for $\underline{\underline{A}}$( 1:$n_b$ , 1:$n_b$ )

Note that while the superscript T in Refs. [54] and [57] denotes the adjoint (transpose and complex conjugate), here it denotes *only* the transpose.

The Level 3 BLAS operations in lines 08 and 13 provide an opportunity for an efficient parallelization of the code, based on MPI, for distributed-memory architectures [56]. However, from the basic differences in the above pseudo-codes which perform the tridiagonalization, it is clear that any implementation of the latter approach would require a complete rewrite of HTD1, and might diminish the overall accessibility of the code in view of the more complex structure of HTD3. This exercise falls outside the scope of this paper and is left for future research, perhaps, within the scope of ongoing (but incomplete) efforts to write massively parallel, multi-precision linear algebra packages with diverse functionality [58].

## References


[1] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University Press, Oxford, UK, 1965.
[2] G. Golub, C.F. van Loan, Matrix Computations, third ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[3] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice Hall, Englewood Cliffs, NJ, 1998.
[4] C.G.J. Jacobi, J. Reine Angew. Math. 30 (1846) 51–94.
[5] J.G.F. Francis, Comput. J. 4 (1961) 265–271.
[6] J.G.F. Francis, Comput. J. 4 (1962) 332–345.
[7] V.N. Kublanovskaya, USSR Comput. Math. Math. Phys. 1 (1963) 637–657.
[8] H. Rutishauser, Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 47–81.
[9] D.S. Watkins, L. Elsner, Lin. Alg. Applic. 143 (1991) 19–47.
[10] H. Xu, SIAM J. Matrix Anal. Appl. 19 (1998) 551–555.
[11] J.H. Wilkinson, Lin. Alg. Applic. 1 (1968) 409–420.
[12] F.R. Gantmakher, The Theory of Matrices, vol. II, Chelsea, New York, 1977.
[13] W.E. Arnoldi, Quart. Appl. Math. 9 (1951) 17–29.
[14] C.M. Bender, D.J. Weir, J. Phys. A 45 (2012) 425303.
[15] G. García-Calderón, . A. Máttar, J. Villavicencio, Phys. Scr. T 151 (2012) 01476.
[16] N. Hatano, G. Ordonez, Int. J. Theor. Phys. 50 (2011) 1105–1115. http://dx.doi.org/10.1007/s10773-010-0576-y.
[17] C. Cohen-Tannoudji, B. Diu, F. Laloë, Quantum Mechanics, vol. 1, first ed., J Wiley & Sons, New York, 1978.
[18] C. Cohen-Tannoudji, B. Diu, F. Laloë, Quantum Mechanics, vol. 2, first ed., J Wiley & Sons, New York, 1978.
[19] S. Salomonson, P. Öster, Phys. Rev. A 40 (1989) 5559–5567.
[20] U.D. Jentschura, Phys. Rev. A 70 (2004) 052108.
[21] U.D. Jentschura, Phys. Rev. A 74 (2006) 062517.
[22] U.D. Jentschura, M. Puchalski, P.J. Mohr, Phys. Rev. A 84 (2011) 064102.
[23] I. Bar-On, V. Ryaboy, SIAM J. Sci. Comput. 18 (1997) 1412–1435.
[24] R. Santra, L.S. Cederbaum, Phys. Rep. 368 (2002) 1–117.
[25] H. Schabauer, C. Pacher, A.G. Sunderland, W.N. Gansterer, Procedia Comput. Sci. 1 (2012) 437–445.
[26] J.H. Noble, M. Lubasch, U.D. Jentschura, Eur. Phys. J. Plus 128 (2013) 93.
[27] J.K. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, vol. I: Theory, Birkhäuser, Boston, 1985.
[28] J.K. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, vol. II: Programs, Birkhäuser, Boston, 1985.
[29] J.K. Cullum, R.A. Willoughby, SIAM J. Math. Anal. 17 (1996) 83–109.
[30] Z. Bai, J. Demmel, Int. J. High Speed Comput. 1 (1989) 97–112.
[31] W. Givens, J. Soc. Indust. Appl. Math. 6 (1958) 26–50.
[32] L.E. Henderson, Testing eigenvalue software, (Ph.D. thesis), University of Arizona, USA (1991), unpublished, available at the URL http://arizona.openrepository.com/arizona/bitstream/10150/185744/1/azu_td_9213693_sip1_m.pdf..
[33] G. Fasshauer, The QR Algorithm in Lecture Notes on *477/577 Numerical Linear Algebra/Computational Mathematics I*, available at the URL http://www.math.iit.edu/~fass/477577_Chapter_11.pdf.
[34] P. Arbenz, The QR Algorithm, Chapter 3 of the Lecture Notes on *Numerical Methods for Solving Large Scale Eigenvalue Problems*, available at the URL http://people.inf.ethz.ch/arbenz/ewp/Lnotes/chapter3pdf.
[35] The gfortran compiler is available at the URL http://hpc.sourceforge.net.
[36] *IBM XL Fortran for AIX Compiler Reference* IBM Corporation, 2012, see the URL http://www-01ibm.com/support/docview.wss?uid=swg27003922{&}aid=1.
[37] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, third ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
[38] U.D. Jentschura, A. Surzhykov, M. Lubasch, J. Zinn-Justin, J. Phys. A 41 (2008) 095302.
[39] U.D. Jentschura, A. Surzhykov, J. Zinn-Justin, Phys. Rev. Lett. 102 (2009) 011601.
[40] U.D. Jentschura, A. Surzhykov, J. Zinn-Justin, Ann. Phys., NY 325 (2010) 1135–1172.
[41] J. Zinn-Justin, U.D. Jentschura, J. Phys. A 43 (2010) 425301.
[42] C.M. Bender, S. Boettcher, P.N. Meisinger, J. Math. Phys. 40 (1999) 2201–2229.
[43] C.M. Bender, S. Boettcher, Phys. Rev. Lett. 80 (1998) 5243–5246.
[44] C.M. Bender, T.T. Wu, Phys. Rev. 184 (1969) 1231–1260.
[45] D.H. Bailey, A portable high performance multiprecision package, NASA Ames Tech. Rep. RNR-90-022.
[46] D.H. Bailey, ACM Trans. Math. Software 19 (1993) 288–319.
[47] D.H. Bailey, ACM Trans. Math. Software 21 (1995) 379–387.
[48] M.H. Macfarlane, Ann. Phys., NY 271 (1999) 159–202.

[49] P. Arbenz, M.E. Hochstenbach, SIAM J. Sci. Comput. 25 (2004) 1655–1673.
[50] G. Garcia-Calderon, R. Peierls, Nuclear Phys. A 265 (1976) 443–460.
[51] G.W.F. Drake, S.P. Goldman, Can. J. Phys. 77 (1999) 835–845.
[52] G.W.F. Drake, M.M. Cassar, R.A. Nistor, Phys. Rev. A 65 (2002) 054501.
[53] See the URL http://crd-legacy.lbl.gov/~dhbailey/mpdist.
[54] J.W. Demmel, M.T. Heath, H.A. van der Vorst, LAPACK Working Note 60, UT CS-93-192 Parallel numerical linear algebra, 1993, see the URL http://www.netlib.org/lapack/lawnspdf/lawn60pdf.
[55] OpenMP Application Program Interface Version 30, OpenMP Architecture Review Board, 2008, see the URL http://www.openmp.org/mp-documents/spec30pdf.
[56] MPI: A Message–Passing Interface Version 30, Message Passing Interface Forum, 2015, see the URL http://www.mpi-forum.org/docs/mpi-30/mpi30-report.pdf.
[57] J. Choi, J.J. Dongarra, D.W. Walker, Numer. Algorithms 10 (1995) 379–399.
[58] J. Martin, MPMPLAPACK: The Massively Parallel Multi- Precision Linear Algebra Package, Talk given at the Seminar on Interactive Parallel Computation in Support of Research in Algebra, Geometry and Number Theory at the Mathematical Science Research Institute in Berkeley, CA 2007, see the URL https://secure.msri.org/communications/vmath/VMathVideos/VideoInfo/2986/show_video.