Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

A NOVEL, BLOCKED ALGORITHM FOR THE REDUCTION TO HESSENBERG-TRIANGULAR FORM*

THIJS STEEL† AND RAF VANDEBRIL†

Abstract. We present an alternative algorithm and implementation for the Hessenberg-triangular reduction, an essential step in the QZ algorithm for solving generalized eigenvalue problems. The reduction step has a cubic computational complexity, and hence, high-performance implementations are compulsory for keeping the computing time under control. Our algorithm is of simple mathematical nature and relies on the connection between generalized and classical eigenvalue problems. Via system solving and the classical reduction of a single matrix to Hessenberg form, we are able to get a theoretically equivalent reduction to Hessenberg-triangular form. As a result, we can perform most of the computational work by relying on existing, highly efficient implementations, which make extensive use of blocking. The accompanying error analysis shows that preprocessing and iterative refinement can be necessary to achieve accurate results. Numerical results show competitiveness with existing implementations.

Key words. generalized eigenvalue problems, Hessenberg-triangular form, blocked algorithms, high-performance computing, GPU acceleration.

AMS subject classifications. 65F15.

1. Introduction.

The generalized eigenvalue problem $Ax = \lambda Bx$, with $A, B \in \mathbb{C}^{n\times n}$, appears in a wide variety of applications. For dense matrices of modest size, the most popular solution method is the QZ algorithm introduced by Moler and Stewart [18], which generalizes the classical QR algorithm. The algorithm consists of two main parts and an optional step to compute the eigenvectors/deflating subspaces if desired.

1. A direct reduction of the pencil to Hessenberg-triangular form via unitary equivalence transformations.
2. An iterative reduction of the Hessenberg-triangular pencil to generalized Schur form, again via unitary equivalence transformations.
3. Calculate, if desired, the eigenvectors via triangular solves.

For an overview of QR and QZ algorithms, we can refer to general textbooks such as Golub and Van Loan [18], Watkins [26], Trefethen and Bau [23], and Demmel [8]. For books entirely dedicated to eigenvalue solvers, we refer to Kressner [17] and Watkins [26]. Recently, a rational version of the QZ algorithm was also introduced by Camps, Meerbergen and Vandebril [7] and Güttel and Berljafa [2].

Bramen, Byers, and Matthias [4,5] contributed significantly to the efficient implementation that is available in LAPACK nowadays for computing eigenvalues of a single matrix via the QR algorithm. Similar improvements were made to the QZ algorithm by Kågström and Kressner [14] and expanded upon by Steel, Camps, Meerbergen, and Vandebril [20]. Consequently, the implementation of the Hessenberg-triangular

*Received by the editors on July 9, 2021. Accepted for publication on October 10, 2022. Handling Editor: Valeria Simoncini. Corresponding Author: Thijs Steel.

†Department of Computer Science, KU Leuven, University of Leuven, 3001 Leuven, Belgium. (thijs.steel@kuleuven.be, raf.vandebril@kuleuven.be), supported by the Research Council KU Leuven, projects C14/16/056 (Inverse-free Rational Krylov Methods: Theory and Applications), OT/14/074 (Numerical algorithms for large scale matrices with uncertain coefficients).

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

reduction featured in LAPACK now dominates the computational cost when calculating a generalized Schur form. In this paper, we therefore focus on an alternative manner of implementing the reduction to Hessenberg-triangular form to achieve better performance.

Highly optimized implementations for the reduction of a single matrix to Hessenberg form exist. We refer to the work of Dongarra, Sorensen, and Hammerling [9]; and Quintana-Ortí and van de Geijn [19]. Their Hessenberg reduction algorithm reduces several columns at a time which is also called a panel reduction. While the Householder reflectors that reduce the panel are being calculated, updates are only applied to the panel. Once the reflectors have been calculated, they are combined into a compact representation that allows the reflectors to be applied to the rest of the matrix using level 3 BLAS.

When reducing a pencil to Hessenberg-triangular form, a panel reduction is much harder. The Hessenberg reduction can be formulated in terms of Householder reflectors that reduce an entire column, whereas the method proposed by Moler and Stewart [18] is formulated in terms of Givens rotations that only reduce a single element. Accumulating these rotations to perform some work in level 3 BLAS is possible, but the fraction of work that can be offloaded this way is smaller than in the Hessenberg reduction. This is the technique proposed by Kågström, Kressner, Quintana-Ortí, and Quintana-Ortí [15]. Reformulating the algorithm in terms of Householder reflectors is also an option and results in the algorithm of Bujanovic, Karlsson and Kressner [6], which is very efficient, but complicated. To clearly illustrate the difference of our approach with the ones mentioned above, we revisit them in detail in Section 2.

This paper is organized as follows. Section 2 discusses the existing implementations for reducing pencils to Hessenberg-triangular form. Section 3 presents the basic algorithm, a manner of efficiently executing iterative refinement and a preprocessing step that is sometimes compulsory to decrease the conditioning. In Section 4, we show that our method is backward stable but can possibly deviate significantly from Hessenberg form. In Section 5, we demonstrate the viability of our approach through numerical experiments.

2. Existing reductions to Hessenberg-triangular form.

Let us briefly discuss the various existing algorithms for reducing a matrix to Hessenberg-triangular form. In all algorithms, we assume without loss of generality that the matrix B is upper triangular.

The classical method presented in Algorithm 1 alternates between eliminating an element in A and eliminating the fill-in in B created by the previous operation. This method only uses level 1 BLAS. The transformations from the left are particularly problematic for their large stride (assuming a matrix stored in column-major order).

Kågström et al. [15] improved the data locality of this algorithm significantly. A simplified version of their method is shown in Algorithm 2. The algorithm works in panels to offload some of the work to level 3 BLAS. Inside the panels, one operates with small elementary transformations, outside the panels one is able to switch to level 3 BLAS. Unfortunately, the panels are quite large, limiting the amount of work that can be offloaded. A version of this algorithm is available in LAPACK.

The last method we consider uses Householder reflectors and is by Bujanovic et al. [6]. Before this paper was released, using Householder reflectors for the reduction to Hessenberg-triangular form was not deemed feasible, because after reducing just one column of A, almost all the upper triangular structure of B is lost. An algorithm based on Householder reflectors would then either have to go through the costly operation of restoring this structure or use smaller Householder reflectors, limiting their computational efficiency.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

Algorithm 1 Moler and Stewart's original reduction [18]

Input: Pencil (A,B) of size $n \times n$, with B upper triangular

```
Q = I, Z = I
for j = 1, 2 ... n − 2 do
    for i = n, n − 1 ... j + 2 do
        Construct a Givens rotation G to make A_{i,j} zero from the left, modifying rows i and i − 1
        Apply G to A and B from the left and apply G* to Q from the right
        Construct a Givens rotation G to make B_{i,i−1} zero from the right, modifying columns i and i − 1
        Apply G to A, B and Z from the right
    end for
end for
```

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

Bujanovic et al. avoid these problems by using opposite Householder reflectors. If a Householder reflector H reduces a vector x (i.e. $Hx = \beta e_1$) and that vector x is the solution of the linear system $Bx = e_1$, then applying BH will have its first column reduced. This is remarkable, because applying a reflector from the right usually reduces a row, not a column. Algorithm 3 illustrates how this technique allows one to perform a Hessenberg-triangular reduction without requiring B to be in upper triangular form. If implemented naïvely, Algorithm 3 will still perform poorly. For computational efficiency, the authors use panels and a compact representation of the reflectors, similar to the work on improving the Hessenberg reduction by Quintana-Ortí et al. [19]. To solve the linear systems, the authors assume B is initially in upper triangular form. While reducing a panel, B will lose this structure, but the authors store UBV, where U and V are the compact representations of the Householder reflectors that should have been applied at that point. This factorization allows for efficient solves. After a panel has been reduced, the reflectors need to be applied to B and the remaining part of A. Doing so using the full representation of the reflectors would destroy the upper triangular structure of B, costing $O(n^3)$ to restore. To avoid this cost, the authors factorize the compact representation of the reflectors (the details of which we will skip). Applying this factorization to B leads to a block upper triangular matrix instead of a full matrix.

To the extent of our knowledge, the last proposed method is currently the fastest one available to reduce a matrix to Hessenberg-triangular form. The algorithm is, however, also complicated and difficult to implement. In this way, it differs greatly from the idea that we will present here. Our algorithm is simple in nature and builds upon existing highly optimized code for solving systems of equations, multiplying matrices, and reducing a matrix to Hessenberg form. The idea is simple, but leads to an algorithm for which an efficient implementation is easy to make.

3. Algorithm. In this section, we introduce our method. The basic method is simple and fast, but is numerically unstable, in the sense that the resulting pair is not exactly of Hessenberg-triangular form, if B is ill-conditioned. This ill-conditioning can lead to large forward errors, but can also result in over- and underflows which is a drawback in comparison to a direct unitary equivalence transformation.

Iterative refinement can be carried out to push the pencil closer and closer to Hessenberg-triangular form. Also, a preprocessing step and a regularization technique can be used in case B is very ill-conditioned or singular. We also note that if B is ill-conditioned and A is not, one can apply the algorithm to the pencil (B, A) and swap the matrices again after having calculated the Schur form.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

Algorithm 2 Panel-wise reduction to Hessenberg-triangular form [15]

Input: Pencil (A,B) of size $n \times n$, with B upper triangular

$Q = I$, $Z = I$
for $k = 1, n_b + 1, \ldots, n$ **do**
  Initialize two sequences of $2n_b \times 2n_b$ matrices $\{U_s, s = 1, 2, \ldots, n_{nb}\}$, $\{V_s, s = 1, 2, \ldots, n_{nb}\}$ as I
  Initialize the $n_b + 1 \times n_b + 1$ matrices $U_{n_{nb}}$ and $V_{n_{nb}}$ as I
  for $j = k, k + 1, \ldots k + n_b - 1$ **do**
    for $i = n, n - 1 \ldots j + 2$ **do**
      Construct a Givens rotation G_i to make $A_{i,j}$ zero from the left, modifying rows i and $i - 1$
    end for
    Apply the sequence $\{G_i, i = n, n - 1 \ldots j + 2\}$ to $B_{i-1:i,k+1:n}$ from the left
    Accumulate the sequence $\{G_i, i = n, n - 1 \ldots j + 2\}$ into the sequence $\{U_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$
    for $i = n, n - 1 \ldots j + 2$ **do**
      Construct a Givens rotation $G_i^{(2)}$ to make $B_{i,i-1}$ zero from the right, modifying columns i and $i - 1$
      Apply $G_i^{(2)}$ to $B_{k+2:i,i-1:i}$ from the right
    end for
    Apply the sequence $\{G_i^{(2)}, i = n, n - 1 \ldots j + 2\}$ to $A_{k+2:n,i-1:i}$ from the right
    Accumulate the sequence $\{G_i^{(2)}, i = n, n - 1 \ldots j + 2\}$ into the sequence $\{V_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$
    If $j < k + n_b - 1$, apply the sequence $\{U_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$ from the left to $A_{k+2:n,j+1}$
  end for
  Apply the sequence $\{U_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$ from the left to $A_{k+2:n,k+n_b:n}$
  Apply the sequence $\{U_s^*, s = 1, 2, \ldots, n_{nb} + 1\}$ from the right to $Q_{1:n,k+2:n}$
  Apply the sequence $\{V_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$ from the right to $A_{1:k+1,k+1:n}$
  Apply the sequence $\{V_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$ from the right to $B_{1:k+1,k+1:n}$
  Apply the sequence $\{V_s, s = n_{nb} + 1, n_{nb}, \ldots, 1\}$ from the right to $Z_{:,k+1:n}$
end for

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

3.1. Basic algorithm. There is a close connection between the generalized and the standard eigenvalue problem (see Golub and Van Loan [12]); as a result reducing a pencil to Hessenberg-triangular form is directly related to the reduction of a single matrix to Hessenberg form. We formalize this in the following theorem.

THEOREM 3.1. *Consider a pencil (A, B), where B is invertible. If Q is a unitary matrix that reduces AB^{-1} to Hessenberg form via a similarity and Z is the conjugate transpose of the unitary factor of the RQ factorization of Q^*B, then $(H, T) = Q^*(A, B)Z$ is a Hessenberg-triangular pencil.*

Proof. Because Z is the conjugate transpose of the unitary factor of the RQ factorization of Q^*B, we conclude that Q^*BZ is upper triangular. By construction, $Q^*AB^{-1}Q$ is in Hessenberg form. If we multiply this matrix on the right with Q^*BZ, we get $Q^*AB^{-1}QQ^*BZ = Q^*AZ$. Since Q^*BZ is upper triangular and multiplying a Hessenberg matrix from the right with an upper triangular matrix does not change the structure of the Hessenberg matrix, Q^*AZ must be Hessenberg. ∎

Based on this theorem, the following algorithm results in a Hessenberg-triangular reduction of a pencil (A, B) in case the matrix B is invertible.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

Algorithm 3 Householder reflector based HT reduction

Input: Pencil (A,B) of size $n \times n$

$Q = I$, $Z = I$
Calculate a Householder reflector H_1 so that $H_1 B_{1:n,1} = \beta_1 e_1$
Apply H_1 to A and B from the left and to Q from the right
for $j = 1, 2 \dots n-2$ **do**
  Calculate a Householder reflector H_2 so that $H_2 A_{j+1:n,j} = \beta_2 e_1$
  Apply H_2 to $A_{j+1:n,j:n}$ and $B_{j+1:n,j:n}$ from the left and to $Q_{1:n,j+1:n}$ from the right
  Solve the linear system $B_{j+1:n,j+1:n} x = e_1$
  Calculate a Householder reflector H_3 so that $H_3 x = \beta_3 e_1$, this also implies that $B_{j+1:n,j+1:n} H_3 e_1 = \beta_4 e_1$
  Apply H_3 to $A_{1:n,j+1:n}$, $B_{1:n,j+1:n}$ and $Z_{1:n,j+1:n}$ from the right
end for

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

Algorithm 4 Simple reduction to Hessenberg-triangular form

Input: Pencil (A,B) of size $n \times n$

$X = AB^{-1}$
Calculate a unitary Q, so that $Q^* X Q$ is Hessenberg
$A = Q^* A$
$B = Q^* B$
Calculate a unitary Z, so that BZ is upper triangular
$A = AZ$
$B = BZ$

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

The greatest strength of this method is that instead of relying on elementary manipulations of the pencil, the method can rely on basic linear algebra routines. Instead of manipulating two matrices with unitary equivalence transformations built from either rotations or reflectors, we can now rely on solving a linear system, a reduction to Hessenberg form, a reduction to upper triangular form and matrix-matrix multiplications. Each of the latter routines has existing, highly efficiently code. Not only does this make the method fast, it also makes it easy to implement, maintain and optimize on different architectures. Unfortunately, the caveat is the accuracy, for which two necessary add-ons are proposed in the next sections.

3.2. Iterative refinement. The calculation of $X = AB^{-1}$ can be sensitive to rounding errors when computed in finite precision arithmetic. This sensitivity is reflected in the conditioning of B, and moreover, if B is singular, X cannot be computed at all.[1] In the next subsection, a preprocessing technique is presented that can be used to adjust the conditioning of the part of B that needs to be inverted to ensure that B is not too ill-conditioned. In this section, we show how, for modestly ill-conditioned B, iterative refinement can be applied in a clever manner to enhance the accuracy.

More precisely, rounding errors in the computation of AB^{-1} imply that Q is not exactly the intended unitary matrix, and as a consequence, $Q^* AZ$ will not exactly be upper Hessenberg, as there will be nonzero

[1] We note that the conditioning of the matrix B for inversion is not the same as the condition of the eigenvalue problem. An eigenvalue problem may have well-conditioned eigenvalues while having an ill-conditioned matrix B.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

elements present below the diagonal. However, as long as B is not too ill-conditioned, the non-Hessenberg part of H will be small and a step of iterative refinement can be executed.

After each iteration, the algorithm checks the sizes of the elements below the diagonal, column by column. If the elements below the subdiagonal in the first columns are small enough, we can treat them as zero and continue the algorithm on the smaller subpencil in the lower-right corner.

Having a subpencil that has some columns less will not have a significant impact on the runtime of the iterations, but it can have a significant impact on the convergence. The Hessenberg-triangular reduction algorithm tends to move infinite eigenvalues to the top; therefore, $B_{k:n,k:n}$ can be significantly better conditioned than B, leading to faster convergence (see also the error analysis in Section 4). We obtain Algorithm 5.

Algorithm 5 Reduction to Hessenberg-triangular form with iterative refinement.

Input: Pencil (A,B) of size $n \times n$

$Q = I$, $Z = I$, $k = 1$
while $k < n$ **do**
  $X = A_{k:n,k:n} B_{k:n,k:n}^{-1}$
  Calculate a unitary Q_c, so that $Q_c^* X Q_c$ is Hessenberg
  $A_{k:n,:} = Q_c^* A_{k:n,:}$
  $B_{k:n,:} = Q_c^* B_{k:n,:}$
  $Q_{:,k:n} = Q_{:,k:n} Q_c$
  Calculate a unitary Z_c, so that $B_{k:n,k:n} Z_c$ is upper triangular
  $A_{:,k:n} = A_{:,k:n} Z_c$
  $B_{:,k:n} = B_{:,k:n} Z_c$
  $Z_{:,k:n} = Z_{:,k:n} Z_c$
  while $\|A_{k+2:n,k}\| \leq \epsilon \|A\|$ **do**
    k = k+1
  end while
end while

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

Suppose that some of the columns of the resulting Hessenberg matrix fulfill our tolerance levels and can be considered in Hessenberg form; the remaining part in the lower-right corner requires another step of iterative refinement. We thus work on matrices that are partially in Hessenberg-triangular form, and it is important to not destroy the part that is already in the correct form.

The first subdiagonal element above the pencil that needs another step of iterative refinement can possibly interfere with unitary updates from the left. This can create a spike as illustrated in Fig. 1. To avoid this, we must ensure that $Qe_1 = e_1$ where Q is the unitary matrix that will carry out the transformation from the left on the remaining subpencil. This means that during an update from the left, Q cannot modify the first row. This constraint can be satisfied trivially!

3.3. Preprocessing. The iterative refinement has its limits. If the pencil has infinite eigenvalues, B will be singular and it will not be possible to calculate X. Additional preprocessing is required in these cases.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

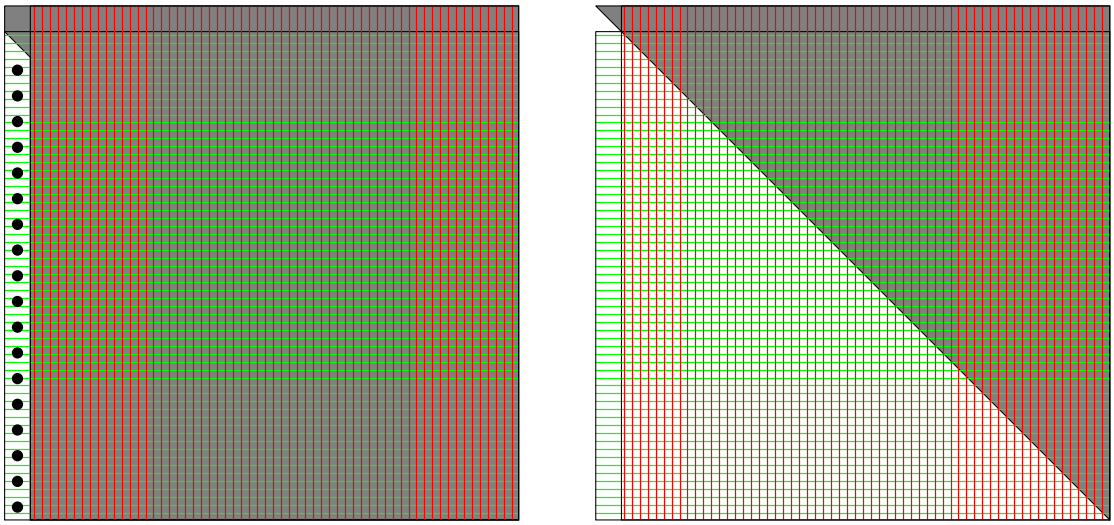


FIG. 1. *Illustration of the unitary updates and the interference with the partial Hessenberg structure. The part of the pencil affected by the updates from the left is illustrated with green horizontal lines, and the part of the pencil affected by the updates from the right is illustrated in red vertical lines. The spike is marked with dots.*

To obtain a reasonably accurate X, we can apply regularization techniques; in our numerical experiments, we have based ourselves on Bujanovic et al. [6], who solve the system $X(B+\Delta) = A$, for a small perturbation Δ. Unfortunately, when there are too many infinite eigenvalues, it is better to perform a preprocessing step, before computing X.

In practical cases, the matrix B is often sparse, and quite some infinite eigenvalues can be identified immediately. The reordering techniques presented by Ward [25] can be used to separate these eigenvalues, resulting in a partial Schur form.

We first compute a factorization of B that moves the infinite eigenvalues to the top. By using a rank revealing RQ factorization (RRRQ), we can separate the ill-conditioned part and the well-conditioned part of the matrix B.[2] In an RRRQ, the rows are pivoted so that in the factorization

$$PB = RQ,$$

with Q unitary, R upper triangular and P a permutation matrix, the entries $r_{i,j}$ of R satisfy

$$|r_{11}| \leq |r_{22}| \leq \cdots \leq |r_{nn}|,$$
$$|r_{kk}| \leq \|R_{j,k:j}\|_2 \text{ for } j = k+1, \ldots, n.$$

If B is rank-deficient, the top left part of R will typically be small in magnitude. For more information of rank revealing QR/RQ factorizations, we refer to Golub and Van Loan [12].

The infinite eigenvalues have been moved to the top, and an alternative reduction method (see Section 2) is used to bring these few columns to Hessenberg-triangular form. Just like in the iterative refinement phase, we end up with a matrix that is partially in Hessenberg-triangular form, and the conditioning of the lower-right remaining block is hopefully small enough so that the iterative refinement scheme converges fast.

[2] An SVD could also be used, but is more expensive.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

Deciding how many columns need to be brought to Hessenberg-triangular form such that the conditioning of the remaining part is acceptable is non-trivial. Reducing more columns implies more computational work performed by any of the slower methods. On the other hand, reducing more columns reduces the condition number of B further, resulting in faster convergence of our algorithm. It is ultimately a tradeoff and depends on how efficient our method is compared to the one chosen to reduce the first columns. In our implementation, we have chosen to stop when the smallest diagonal element in B is smaller than $\epsilon\|B\|$, where ϵ is the machine precision. The full algorithm including preprocessing and iterative refinement is described in Algorithm 6.

Algorithm 6 Full Hessenberg-triangular reduction algorithm with preprocessing and iterative refinement.

Input: Pencil (A,B) of size $n \times n$

$k = 1$
Calculate the RRRQ of B: PBZ_c
$A = PBZ_c$
$B = PBZ_c$
$Q = P^*$
$Z = Z_c$
while $B_{k,k} < \epsilon\|B\|$ **do**
  Reduce a single column of the pencil using an appropriate HT reduction algorithm
  $k = k + 1$
end while
while $k < n$ **do**
  Choose a matrix Δ so that $B_{k:n,k:n} + \Delta$ is nonsingular
  $X = A_{k:n,k:n}(B_{k:n,k:n} + \Delta)^{-1}$
  Calculate Q_c, so that $Q_c^* X Q_c$ is Hessenberg
  $A_{k:n,:} = Q_c^* A_{k:n,:}$
  $B_{k:n,:} = Q_c^* B_{k:n,:}$
  $Q_{:,k:n} = Q_{:,k:n} Q_c$
  Calculate Z_c, so that $B_{k:n,k:n} Z_c$ is upper triangular
  $A_{:,k:n} = A_{:,k:n} Z_c$
  $B_{:,k:n} = B_{:,k:n} Z_c$
  $Z_{:,k:n} = Z_{:,k:n} Z_c$
  while $\|A_{k+2:n,k}\| \leq \epsilon\|A\|$ **do**
    k = k+1
  end while
end while

Output: Hessenberg-triangular pencil (A,B) and unitary matrices Q and Z

Finally, we note that in many cases, see also the numerical experiments, we do not need to execute the preprocessing step. So, depending on prior knowledge of the input data, one might decide to skip this possibly time-consuming step.

4. Error analysis. In this section, we study how rounding errors propagate through the algorithm. First, we will prove that each iteration of Algorithm 6 is backward stable in Theorem 4.1. However, this theorem does not guarantee that the pencil is in Hessenberg-triangular form. Second, we will bound the deviation from Hessenberg-triangular form after one iteration in Theorem 4.2. Third, we will prove that if the iterative refinement scheme converges, the entirety of Algorithm 6 is backward stable in Theorem 4.3

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

THEOREM 4.1 (Normwise backward stability of a single iteration of Algorithm 6). *Given a pencil (A, B). Let $(\tilde{H}, \tilde{T})$ be the result of one iteration of Algorithm 6 in finite precision arithmetic. If we assume that each step of Algorithm 6 is implemented using backward stable methods, then there exist exactly unitary matrices Q and Z so that the resulting pencil $(\tilde{H}, \tilde{T})$ satisfies:*

$$Q(\tilde{H}, \tilde{T})Z^* = (A + \Delta_A, B + \Delta_B), \tag{4.1}$$

with $\tilde{T}$ upper triangular, $\Delta_A \leq c\,\epsilon\,\|A\| + O(\epsilon^2)$, $\Delta_B \leq c\,\epsilon\,\|B\| + O(\epsilon^2)$, ϵ the machine precision and c a modest constant, depending on the size of the problem. This holds for any submultiplicative matrix norm.

Theorem 4.1 guarantees that the computed decomposition is backward stable, but it does not guarantee that $(\tilde{H}, \tilde{T})$ is in Hessenberg-triangular form. More specifically, $\tilde{H}$ is not guaranteed to be in Hessenberg form. In Theorem 4.2, we show that $\tilde{H}$ is Hessenberg up to a term that depends on the condition number of B. This is why the preprocessing step to reduce the ill-conditioning of B and the iterative refinement to remove the remaining error below the subdiagonal of the matrix $\tilde{H}$ are often compulsory.

THEOREM 4.2 (Bound on the perturbation after one iteration). *Given a pencil (A, B), with B numerically nonsingular (i.e. $\kappa(B) < \epsilon^{-1}$). If one iteration of Algorithm 6 is executed in finite precision arithmetic, then (A, B) is transformed into a pencil $(\tilde{H}, \tilde{T})$, where the matrix $\tilde{T}$ is upper triangular and $\tilde{H}$ can be written as the sum of a Hessenberg matrix H plus a perturbation Δ_H, where we can bound the norm of Δ_H as follows:*

$$\|\Delta_H\| \leq c\,\epsilon\,\|A\|\kappa(B) + O(\epsilon^2), \tag{4.2}$$

with ϵ the machine precision and c a modest constant, depending on the size of the problem, and this holds for any submultiplicative matrix norm.

For the iterative refinement scheme to converge, Δ_H must satisfy $\|\Delta_H\| \leq c\epsilon\|A\|$ after some iterations. Theorem 4.2 does not guarantee this as the bound for Δ_H does not decrease after a refinement step. However, we can still prove that the iterative refinement scheme will converge. Each step will accurately reduce the first column of the subpencil. Because B is upper triangular, the first column of X is always a multiple of the first column of A. The calculation of this column is not sensitive to roundoff even if B is ill-conditioned. As a result, Q will accurately reduce the first column of A. Even if the calculation of X overflows, we can always apply a large enough perturbation so that the regularized solution $A(B+\Delta)^{-1}$ can be calculated, for example, $\Delta = I - B$. Because each iteration accurately reduces at least one column, we can guarantee that in the worst case, the refinement scheme will converge in n steps. More interesting is that after k columns have been accurately reduced, we only need to consider the submatrix $B_{k:n,k:n}$, which tends to be better conditioned. In the next section, the numerical experiments will indicate that the algorithm converges fast for pencils that do not have many infinite eigenvalues. Unfortunately, we were unable to prove a sharper bound on convergence.

THEOREM 4.3 (Normwise backward stability of Algorithm 6). *Given a pencil (A, B). Let $(\tilde{H}, \tilde{T})$ be the result Algorithm 6 in finite precision arithmetic and assume that the algorithm has converged in the sense that there is an exactly Hessenberg matrix H so that $\|H - \tilde{H}\| \leq c_{tol}\epsilon\|A\|$. If we assume that each step of Algorithm 6 is implemented using backward stable methods, then there exist exactly unitary matrices Q and Z so that:*

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

$$Q(H,\tilde{T})Z^* = (A+\Delta_A, B+\Delta_B), \tag{4.3}$$

with $\tilde{T}$ upper triangular, $\Delta_A \leq c\,\epsilon\,\|A\| + O(\epsilon^2)$, $\Delta_B \leq c\,\epsilon\,\|B\| + O(\epsilon^2)$, ϵ the machine precision and c a modest constant, depending on the size of the problem. This holds for any submultiplicative matrix norm.

We conclude this section by proving Theorems 4.1, 4.2, and 4.3.

Proof. Throughout the proof, we use c's to indicate small constants that may depend on the size of the problem, but are independent of the matrices themselves.

Let us examine the deviation Δ_H from being Hessenberg. We start by calculating $X = AB^{-1}$, and we get $\tilde{X}$. This is essentially a series of linear system solves $(e_i^T X)B = (e_i^T A)$, involving the rows of A and X and the full matrix B. By assumption, the backward error for each linear system equals[3][4] ($i = 1, \ldots, n$):

$$e_i^T \tilde{X}(B + E_{B,i}) = e_i^T A - E_{A,i}, \tag{4.4}$$

where we can bound the perturbations as $\|E_{B,i}\| \leq c_{B,i}\,\epsilon\,\|B\| + O(\epsilon^2)$ and $\|E_{A,i}\| \leq c_A\,\epsilon\,\|e_i^T A\| + O(\epsilon^2)$. We remark that each system i has its individual perturbation matrix $E_{B,i}$ and vector $E_{A,i}$. Define E_A such that $e_i^T E_A = E_{A,i}$. Rewriting (4.4) we get the following row-wise forward error result for X:

$$e_i^T X = e_i^T AB^{-1} = e_i^T \tilde{X} + \left(e_i^T \tilde{X} E_{B,i} + e_i^T E_A\right) B^{-1}, \quad \text{with } i = 1, \ldots, n.$$

We introduce a matrix E_B so that row i of E_B equals row i of $\tilde{X}E_{B,i}$, thus $e_i^T E_B = e_i^T(\tilde{X}E_{B,i})$, for $i = 1, \ldots, n$. As a result, the following forward error on X is obtained:

$$X = AB^{-1} = \tilde{X} + (E_B + E_A)B^{-1}. \tag{4.5}$$

We already have $\|E_A\| \leq c_A\,\epsilon\,\|A\| + O(\epsilon^2)$. To bound $\|E_B\|$, we must bound $\|\tilde{X}\|$. Since $\|\tilde{X} - X\|/\|X\| \leq c_0\epsilon\kappa(B) + O(\epsilon^2)$ and by assuming $\kappa(B) < \epsilon^{-1}$ we get $\|\tilde{X} - X\| < c_1\|X\| + O(\epsilon^2)$ (w.l.g. $\epsilon < 1$). As a consequence, we get $\|\tilde{X}\| \leq \|X\| + \|\tilde{X} - X\| < (1 + c_1)\|X\| \leq (1 + c_1)\|A\|\|B^{-1}\|$. Making use of this bound on $\|\tilde{X}\|$ we get $\|E_B\| \leq c_2\,\epsilon\,\|B\|\|\tilde{X}\| + O(\epsilon^2) \leq c_B\,\epsilon\,\kappa(B)\,\|A\| + O(\epsilon^2)$. The condition number of B is already introduced in the computation of X. We will see, however, that the forward error will stay of the same order.

Next, we will transform our computed $\tilde{X}$ to Hessenberg form. If a backward stable Hessenberg reduction is used, we end up with a matrix $\tilde{Q}$ that is almost unitary and a Hessenberg matrix $\tilde{H}_{\tilde{X}}$. The resulting $\tilde{Q}$ and $\tilde{H}_{\tilde{X}}$ satisfy the relation (see Trefethen and Bau Theorem 26.1 [23] and Higham Section 18.4 [13])

$$\begin{aligned} \tilde{H}_{\tilde{X}} + E_{\tilde{H}} &= \tilde{Q}^* \tilde{X} \tilde{Q}, \text{ with } \|E_{\tilde{H}}\| \leq c_3\,\epsilon\,\|\tilde{X}\| + O(\epsilon^2) \leq c_{\tilde{H}}\,\epsilon\,\|A\|\|B^{-1}\| + O(\epsilon^2), \\ \tilde{Q} &= Q + \Delta_Q, \text{ with } QQ^* = I\ , \|\Delta_Q\| \leq c_{\tilde{Q}}\,\epsilon\, + O(\epsilon^2), \\ \tilde{Q}\tilde{Q}^* &= I - E_{\tilde{Q}}, \text{ with } \|E_{\tilde{Q}}\| \leq c_{\tilde{Q}}\,\epsilon\, + O(\epsilon^2). \end{aligned} \tag{4.6}$$

We remark that we only state that $\tilde{Q}$ is close to being unitary; $\tilde{Q}$ could be far from the exact Q that would reduce X to Hessenberg form.

Next, we compute Z as the conjugate transpose of the unitary factor in the RQ decomposition of the matrix $\tilde{Q}^*B$. The multiplication with $\tilde{Q}$ as well as computing the RQ decomposition are backward stable

[3] We deliberately added a minus sign to $E_{A,i}$ to simplify forthcoming deductions.
[4] In this proof we focus on nonsingular B, so we ignore the regularization term Δ.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

operations, and we end up with a computed upper triangular matrix $\tilde{T}$ satisfying the following relation (see Higham Sections 18.3 and 18.4 [13]):

$$\begin{aligned} \tilde{T} + E_{\tilde{T}} &= \tilde{Q}^* B \tilde{Z}, \text{ with } \|E_{\tilde{T}}\| \leq c_{\tilde{T}}\, \epsilon\, \|B\| + O(\epsilon^2), \\ \tilde{Z} &= Z + \Delta_Z, \text{ with } ZZ^* = I\ , \|\Delta_Z\| \leq c_Z\, \epsilon\ + O(\epsilon^2), \\ \tilde{Z}\tilde{Z}^* &= I - E_{\tilde{Z}}, \text{ with } \|E_{\tilde{Z}}\| \leq c_{\tilde{Z}}\, \epsilon\ + O(\epsilon^2). \end{aligned} \tag{4.7}$$

The computed $\tilde{Q}$ and $\tilde{Z}$ are then used to execute an equivalence transformation on the matrix A. As a result of these multiplications, we obtain

$$\tilde{H} = \tilde{Q}^* A \tilde{Z} + E_M, \text{ with } \|E_M\| \leq c_M\, \epsilon\, \|A\| + O(\epsilon^2). \tag{4.8}$$

We can now combine all previous bounds to deduce how far $\tilde{H}$ is from being Hessenberg. Rewriting the equation for $\tilde{H}$ by inserting $B^{-1}B$ and using Equation (4.5) results in

$$\begin{aligned} \tilde{H} &= \tilde{Q}^* A \tilde{Z} + E_M = \tilde{Q}^* A B^{-1} B \tilde{Z} + E_M \\ &= \tilde{Q}^* \tilde{X} B \tilde{Z} + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M. \end{aligned} \tag{4.9}$$

Inserting $I = \tilde{Q}\tilde{Q}^* + E_{\tilde{Q}}$ in the first term in equation (4.9) between $\tilde{X}$ and B leads to

$$\begin{aligned} \tilde{H} &= \tilde{Q}^* \tilde{X} \tilde{Q}\ \tilde{Q}^* B \tilde{Z} + \tilde{Q}^* \tilde{X} E_{\tilde{Q}} B \tilde{Z} + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M \\ &= \tilde{Q}^* \tilde{X} \tilde{Q}\ \tilde{Q}^* B \tilde{Z} + E_Q + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M, \end{aligned}$$

where $E_Q = \tilde{Q}^* \tilde{X} E_{\tilde{Q}} B \tilde{Z}$ satisfies $\|E_Q\| \leq c_Q\, \epsilon\, \kappa(B) \|A\| + O(\epsilon^2)$.

We use (4.7) to introduce $\tilde{T}$ and (4.6) to introduce $\tilde{H}_{\tilde{X}}$:

$$\begin{aligned} \tilde{H} &= \left(\tilde{H}_{\tilde{X}} + E_{\tilde{H}}\right)\ \left(\tilde{T} + E_{\tilde{T}}\right) + E_Q + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M \\ &= \tilde{H}_{\tilde{X}} \tilde{T} + \underbrace{E_{\tilde{H}} \tilde{T} + \tilde{H}_{\tilde{X}} E_{\tilde{T}} + E_Q + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M + O(\epsilon^2)}_{=\Delta_H} \end{aligned}$$

The matrix $\tilde{H}_{\tilde{X}}\tilde{T}$ is upper Hessenberg since the multiplication on the right with an upper triangular matrix $\tilde{T}$ does not perturb the Hessenberg structure of $\tilde{H}_{\tilde{X}}$. As a result, the deviation Δ_H from upper Hessenberg form can be bounded as:

$$\begin{aligned} \|\Delta_H\| &= \|E_{\tilde{H}} \tilde{T} + \tilde{H}_{\tilde{X}} E_{\tilde{T}} + E_Q + \tilde{Q}^* (E_A + E_B)\tilde{Z} + E_M + O(\epsilon^2) \\ &\leq c_4\, \epsilon\, \|A\| \kappa(B) + c_5 \epsilon\, \|A\| + O(\epsilon^2) \\ &\leq c\, \epsilon\, \|A\| \kappa(B) + O(\epsilon^2). \end{aligned}$$

This concludes the proof of Theorem 4.2.

Next, we prove the backward stability of the decomposition. Combining equations (4.6), (4.7) and (4.8), we get that after an iteration

$$(\tilde{H}, \tilde{T}) = (Q + \Delta_Q)^* (A + E_m, B - E_{\tilde{T}})(Z + \Delta_Z).$$

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

After some manipulation, this becomes

$$Q(\tilde{H},\tilde{T})Z^* = (A + \underbrace{QE_mZ^* + Q\Delta_Q^*A + A\Delta_ZZ^* + O(\epsilon^2)}_{=\Delta_A}, B + \underbrace{(-QE_{\tilde{T}}Z^*) + Q\Delta_Q^*B + B\Delta_ZZ^* + O(\epsilon^2)}_{=\Delta_B}).$$

It is easy to see that Δ_A and Δ_B satisfy the bounds stated in Theorem 4.1, completing the proof of that theorem.

Finally, we prove Theorem 4.3, which is a direct consequence of Theorem 4.1. Let us use superscripts to differentiate matrices from different iterations, so that $(\tilde{H}^{(i)},\tilde{T}^{(i)})$ is the resulting pencil after i iterations. Since we have already proven that each iteration is backward stable, it is easy to see that there exist unitary matrices $Q^{(i)}$ and $Z^{(i)}$ so that

$$Q^{(i)}(\tilde{H}^{(i)},\tilde{T}^{(i)})Z^{(i)*} = (A + \Delta_A^{(i)}, B + \Delta_B^{(i)}),$$

with $\Delta_A \leq c\,\epsilon\,\|A\| + O(\epsilon^2)$, $\Delta_B \leq c\,\epsilon\,\|B\| + O(\epsilon^2)$. Consider the exactly Hessenberg matrix $H^{(i)} = \tilde{H}^{(i)} - \Delta_H^{(i)}$, which satisfies

$$Q^{(i)}(H^{(i)},\tilde{T}^{(i)})Z^{(i)*} = (A + \Delta_A^{(i)} + Q^{(i)}\Delta_H^{(i)}Z^{(i)*}, B + \Delta_B^{(i)}).$$

By assumption, the Algorithm has converged and so $\|\Delta_H\| < c_{tol}\epsilon\|A\|$. This lets us bound the perturbation on A as $\|\Delta_A^{(i)} + Q^{(i)}\Delta_H^{(i)}Z^{(i)*}\| \leq (c^{(i)} + c_{tol})\epsilon\|A\|$, which concludes the proof of Theorem 4.3. ∎

5. Numerical experiments. In this section, we illustrate the performance of the algorithm. We compare the following implementations:

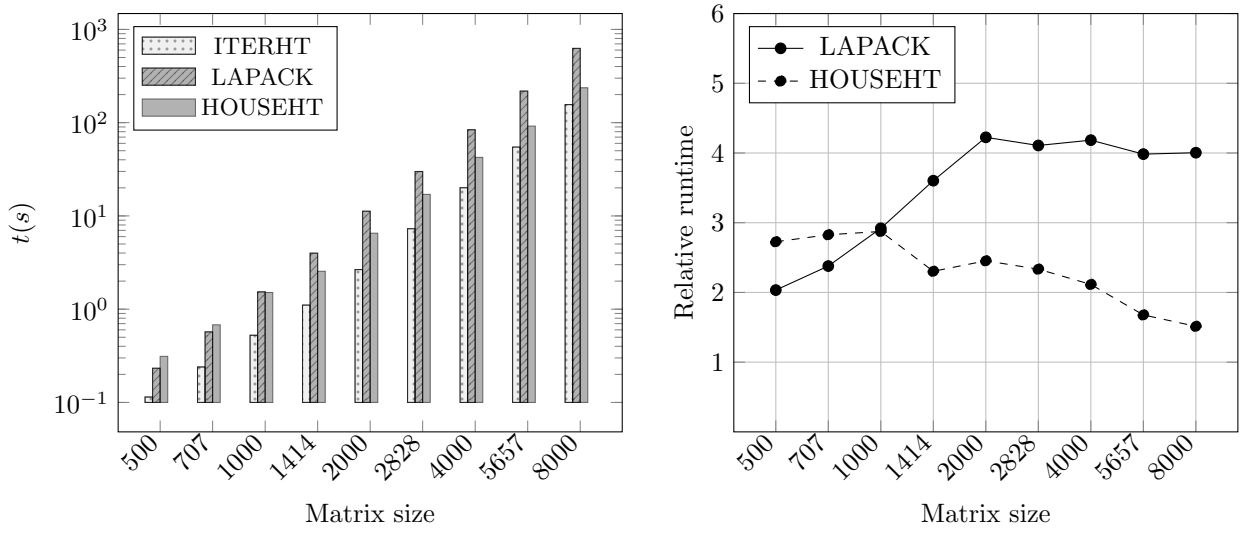
- ITERHT: a Fortran implementation of the method presented in this paper.
- ITERHT$_{\text{GPU}}$: a version of ITERHT modified to use the GPU.
- XGGHD3: The blocked Hessenberg-triangular reduction by Kågström et al. [15] as implemented in LAPACK 3.9 [1] (Algorithm 2).
- HOUSEHT: Householder reflector based reduction by Bujanovic et al. [6] (Algorithm 3).

We use two machines to test the performance: a compute server with two Intel Xeon E5-2650 v2 CPUs with 8 cores each and 62Gb of RAM, and a laptop with an Intel i7-8750H CPU with 6 cores, 16Gb of RAM and an NVIDIA 1060 GPU. All the codes (including XGGHD3, but not the rest of LAPACK or BLAS) are compiled using GCC version 7.5.0 with optimization option -O3 and linked with MKL [24] and MAGMA [22]. Unless mentioned explicitly, no preprocessing is applied.

5.1. Tests on randomly generated pencils. For the first experiment, we apply the algorithms to pencils with entries drawn from a uniform distribution between 0 and 1. Such a random matrix is expected to be nonsingular, so ITERHT should perform well. Results are shown in Figs. 2 and 3. On both machines, our method is consistently the fastest for all pencil sizes. However, on the laptop, the speedup of ITERHT compared to the other method is less impressive. This could be caused by many different factors, but we suspect it is because some of the speedup can be explained by our method being able to more efficiently use many cores, which the laptop has less of.

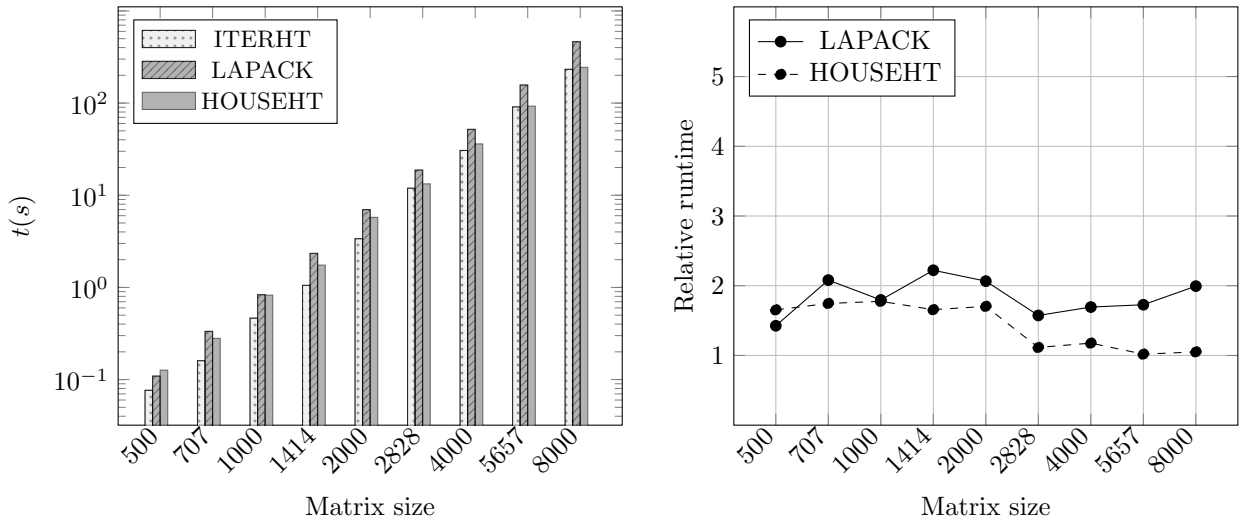
5.2. Examples from matrix market. For the next experiment, we tested some matrices that are related to physical problems on the compute server. These consist of the *cavity* and *obstacle flow* pencils

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.



(a) Runtime (b) Relative runtime

FIG. 2. *Performance of the different algorithms on the compute server for randomly generated pencils of different sizes n. The left figure shows the runtime of the different algorithms. The right figure shows the relative runtime of LAPACK and HOUSEHT compared to the runtime of ITERHT. This is the same as the speedup of ITERHT when compared with the other algorithms.*



(a) Runtime (b) Relative runtime

FIG. 3. *Performance of the different algorithms on the laptop for randomly generated pencils of different sizes n. The left figure shows the runtime of the different algorithms. The right figure shows the relative runtime of LAPACK and HOUSEHT compared to the runtime of ITERHT.*

generated with IFISS [10,11], and two pencils from Matrix market [3] originating from the MHD collection and the *rail* pencil from the Oberwolfach benchmark collection [16]. Results are shown in Table 1. We note that some of the matrices can be considered numerically singular, with which one could expect our method to perform poorly. However, the experiment shows fast convergence.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

TABLE 1

Performance on the different algorithms on the compute server for pencils from practice. The columns denote the name of the problem (Problem), the number of rows/columns in the matrices (Size), the condition of the second matrix of the pencil ($\kappa(B)$), the number of refinement step that were executes during the ITERHT algorithm and finally the execution time of ITERHT ($t_{iterht}(s)$), LAPACK ($t_{lapack}(s)$) and HOUSEHT ($t_{househt}(s)$) respectively. Results between parenthesis are for the single precision implementation.

Problem	Size	$\kappa(B)$	Refine	$t_{iterht}(s)$	$t_{lapack}(s)$	$t_{househt}(s)$
Cavity flow	2467	1.9e18	1 (1)	4.57 (2.86)	21.3 (15.6)	12.6 (7.50)
Obstacle flow	2488	1.8e3	0 (1)	2.38 (2.90)	21.9 (15.9)	12.0 (7.28)
Mhd 3200	3200	1.6e13	1 (1)	10.1 (2.72)	42.1 (32.9)	27.8 (15.8)
Mhd 4800	4800	8.2e13	1 (0)	34.2 (9.45)	137 (102)	81.2 (44.0)
Rail	5177	3.5e2	0 (3)	21.1 (31.6)	165 (127)	79.5 (50.0)

5.3. Scalability. For the third experiment, we vary the number of threads MKL is allowed to use. As the calls to MKL are the only source of parallelism in our code, this should give a good indication of the scalability of our algorithm. We perform this test on the same randomly generated pencil as in the first experiment with $n = 8000$, Fig. 4 shows the results. While some parts of ITERHT scale very well (especially the application of the reflectors and the solution of the upper triangular system), the Hessenberg reduction does not. As a whole, ITERHT scales better than HOUSEHT and both methods scale a lot better than LAPACK. This makes our method of particular interest for modern architectures which typically have more cores.

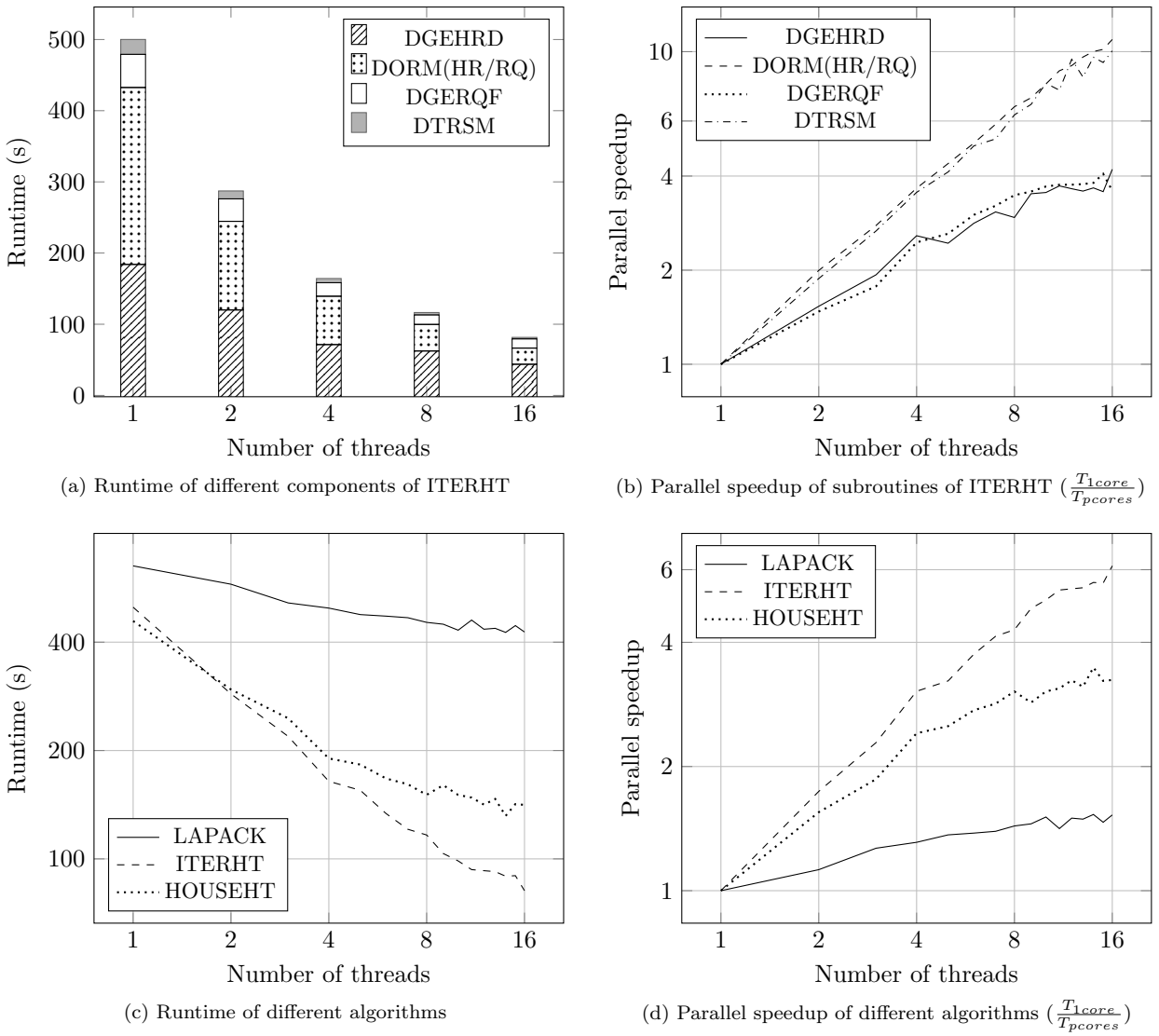
5.4. GPU implementation. As shown in the previous experiments, much of the performance of our new algorithm can be attributed to its ability to utilize many cores effectively. Unfortunately, at the time of writing, many consumer-grade CPUs do not have enough cores to fully utilize this scalability. However, many of these systems have a GPU and the Hessenberg reduction scales very well on GPUs [21]. For the final experiment, we implemented a version of ITERHT that uses MAGMA instead of LAPACK for most of the calculations and tested it on the random pencils of experiment 1. Results of the CPU and GPU implementations on the laptop are shown in Fig. 5. These results seem to indicate that in single precision this implementation performs very well, achieving a speedup of 4 over the CPU implementation. The specific GPU we used is (like most GPUs) designed for single precision and cannot achieve the same speed in double precision.

5.5. Saddle point problem. Inspired by Bujanovic et al. [6], we stress test our algorithm using saddle point problems. These are problems of the form:

$$A = \begin{bmatrix} X & Y \\ Y^T & 0 \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix},$$

where $A, B \in \mathbb{R}^{n\times n}$. $X \in \mathbb{R}^{(n-k)\times(n-k)}$ and $Y \in \mathbb{R}^{(n-k)\times k}$ are randomly generated. And I is the $(n-k)\times(n-k)$ identity matrix. This pencil contains (at least) k infinite eigenvalues, so by varying k, we can make this pencil harder to solve for our algorithm. Figure 6 shows the results of applying our algorithm to these saddle point problems. If no preprocessing is applied, the performance clearly suffers from the presence of the infinite eigenvalues even though the number of infinite eigenvalues is relatively small. The number of required refinement steps seems to scale linearly with the number of infinite eigenvalues. This is likely

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.



(a) Runtime of different components of ITERHT

(b) Parallel speedup of subroutines of ITERHT ($\frac{T_{1core}}{T_{pcores}}$)

(c) Runtime of different algorithms

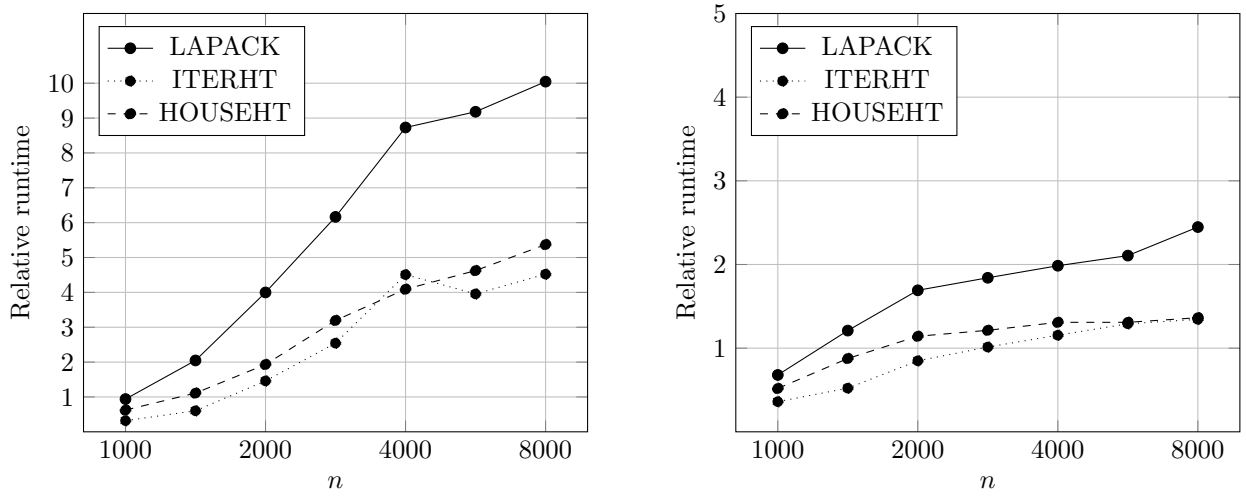
(d) Parallel speedup of different algorithms ($\frac{T_{1core}}{T_{pcores}}$)

FIG. 4. *Performance on the compute server for randomly generated pencils of size* 8000. *The top two plots show the breakdown of the runtime of ITERHT into its largest components and their parallel speedup. When using one core, the most expensive subroutines are DGEHRD (the Hessenberg reduction of X) and DORM(HR/RQ) (the application of the reflectors to the matrices). When using multiple cores, DGERQF (the RQ decomposition) also becomes an important part of the runtime. The bottom two plots show the runtime and parallel speedup of different algorithms.*

because the algorithm relies on the guaranteed reduction of a single column to remove the infinite eigenvalues one at a time. When we do run the preprocessing step, however, no refinement steps are necessary.

6. Conclusion. We presented a new algorithm for the reduction to Hessenberg-triangular form. By utilizing the link between generalized and standard eigenvalue problems, we can use readily available software to perform most of the computational tasks. This leads to a high degree of computational efficiency and keeps the code simple. We implemented the new algorithm and experimentally verified that the algorithm

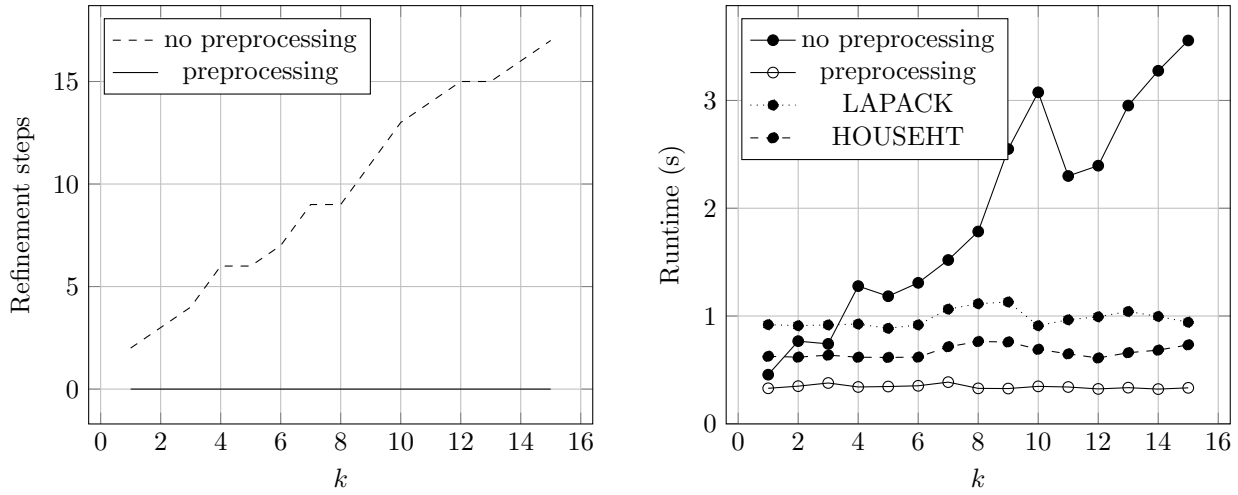
Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.



(a) Single precision

(b) Double precision

FIG. 5. *Performance of the algorithm relative to $ITERHT_{GPU}$ on the laptop for randomly generated pencils of different sizes n.*



(a) The required number of refinement steps (excluding the initial step) for ITERHT

(b) The runtime of ITERHT with and without preprocessing and of LAPACK and HOUSEHT without preprocessing. The timings of ITERHT with preprocessing include the time it takes for the preprocessing.

FIG. 6. *The influence of the number of infinite eigenvalues k on the runtime and number of refinement steps when reducing saddle point problem of size 1000×1000. The experiment is performed both with and without preprocessing. With preprocessing, no refinement steps are needed. Without preprocessing, the number of refinement steps grows linearly with the amount of infinite eigenvalues.*

converges in only a few iterations and is faster than state-of-the-art algorithms when the pencil does not have infinite eigenvalues. However, the algorithm is not suited for pencils that have a large number of infinite eigenvalues. For such pencils, we need to use preprocessing to remove the problematic eigenvalues. We were unable to find a sharp bound on the convergence rate, and we hope that this will be solved in future research.

Electronic Journal of Linear Algebra, ISSN 1081-3810
A publication of the International Linear Algebra Society
Volume 38, pp. 712-728, October 2022.

Acknowledgements. The authors thank Mirko Myllykoski and Karl Meerbergen for useful discussions regarding the implementation and error analysis of the algorithm. We also thank Bujanovic, Karlsson, and Kressner for providing us their implementation to compare against.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*, Third edition. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.

[2] M. Berljafa and S. Güttel. Generalized rational Krylov decompositions with an application to rational approximation. *SIAM J. Matrix Anal. Appl.*, 36(2):894–916, 2015.

[3] R.F. Boisvert, R. Pozo, K. Remington, R.F. Barrett, and J.J. Dongarra. Matrix market: A web resource for test matrix collections. In: *Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software: Assessment and Enhancement*, 125–137. Chapman & Hall, Ltd, London, UK, 1997.

[4] K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. Part I: Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix Anal. Appl.*, 23(4):929–947, 2002.

[5] K. Braman, R. Byers, and R. Mathias. The multishift QR algorithm. Part II: Aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 23(4):948–973, 2002.

[6] Z. Bujanovic, L. Karlsson, and D. Kressner. A Householder-based algorithm for Hessenberg-triangular reduction. *SIAM J. Matrix Anal. Appl.*, 39(3):1270–1294, 2018.

[7] D. Camps, K. Meerbergen, and R. Vandebril. A rational QZ method. *SIAM J. Matrix Anal. Appl.*, 40(3):943–972, 2019.

[8] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, Pennsylvania, USA, 1997.

[9] J.J. Dongarra, D.C. Sorensen, and S.J. Hammarling. Block reduction of matrices to condensed forms for eigenvalue computations. *J. Computat. Appl. Math.*, 27(1–2):215–227, 1989.

[10] H. Elman, A. Ramage, and D. Silvester. Algorithm 866: IFISS, a Matlab toolbox for modeling incompressible flow. *ACM Trans. Math. Softw.*, 33:2–14, 2007.

[11] H. Elman, A. Ramage, and D. Silvester. IFISS: A computational laboratory for investigating incompressible flow problems. *SIAM Rev.*, 56:261–273, 2014.

[12] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore and London, 2013.

[13] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, 1996.

[14] B. Kågström and D. Kressner. Multishift variants of the QZ algorithm with aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 29(1):199–227, 2007.

[15] B. Kågström, D. Kressner, E.S. Quintana-Ortí, and G. Quintana-Orti. Blocked algorithms for the reduction to Hessenberg-triangular form revisited. *BIT Numer. Math.*, 48(3):563–584, 2008.

[16] J.G. Korvink and E.B. Rudnyi. Oberwolfach benchmark collection. In: P. Benner, D.C. Sorensen, and V. Mehrmann (editors), *Dimension Reduction of Large-Scale Systems*, 311–315, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[17] D. Kressner. *Numerical Methods for General and Structured Eigenvalue Problems*, vol. 46. LNCSE. Springer, Berlin Heidelberg, 2005.

[18] C.B. Moler and G.W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10(2):241–256, 1973.

[19] G. Quintana-Ortí and R. van de Geijn. Improving the performance of reduction to Hessenberg form. *ACM Trans. Math. Softw. (TOMS)*, 32(2):180–194, 2006.

[20] T. Steel, D. Camps, K. Meerbergen, and R. Vandebril. A multishift, multipole rational QZ method with aggressive early deflation. *SIAM J. Matrix Anal. Appl.*, 42(2):753–774, 2021.

[21] S. Tomov and J. Dongarra. Accelerating the reduction to upper Hessenberg form through hybrid GPU-based computing. Technical report, LAPACK Working Note 219, 2009.

[22] S. Tomov, J. Dongarra, and M. Baboulin. Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Comput.*, 36(5-6):232–240, June 2010.

[23] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, Pennsylvania, USA, 1997.

[24] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Y. Wang. Intel math kernel library. In *High-Performance Computing on the Intel® Xeon Phi™*, 167–188. Springer, Cham, 2014.

[25] R.C. Ward. Balancing the generalized eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 2(2):141–152, 1981.

[26] D.S. Watkins. *Fundamentals of Matrix Computations*, Third edition. Pure and Applied Mathematics. John Wiley & Sons, Inc., New York, 2010.