The roots of a monic polynomial expressed in a Chebyshev basis are known to be the eigenvalues of the so-called colleague matrix, which is a Hessenberg matrix that is the sum of a symmetric tridiagonal matrix and a rank-1 matrix. The rootfinding problem is thus reformulated as an eigenproblem, making the computation of the eigenvalues of such matrices a subject of significant practical importance. In this report, we describe an $O(n^2)$ explicit structured QR algorithm for colleague matrices and prove that it is componentwise backward stable, in the sense that the backward error in the colleague matrix can be represented as relative perturbations to its components. A recent result of Noferini, Robol, and Vandebril shows that componentwise backward stability implies that the backward error $\delta c$ in the vector $c$ of Chebyshev expansion coefficients of the polynomial has the bound $\|\delta c\| \lesssim \|c\| u$, where $u$ is machine precision. Thus, the algorithm we describe has both the optimal backward error in the coefficients and the optimal cost $O(n^2)$. We illustrate the performance of the algorithm with several numerical examples.

# A Provably Componentwise Backward Stable $\mathbf{O(n^2)}$ QR Algorithm for the Diagonalization of Colleague Matrices

K. Serkh[†◇], V. Rokhlin[‡⊕],
University of Toronto NA Technical Report
February 24, 2021

◇ This author's work was supported in part by the NSERC Discovery Grants RGPIN-2020-06022 and DGECR-2020-00356.
⊕ This author's work was supported in part under ONR N00014-18-1-2353 and NSF DMS-1952751.

† Dept. of Math. and Computer Science, University of Toronto, Toronto, ON M5S 2E4
‡ Dept. of Mathematics, Yale University, New Haven, CT 06511

# Contents



# 1 Introduction

The problem of finding the roots of the polynomial

$$p(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} + x^n \tag{1}$$

is one of the oldest and most classical problems in mathematics. Countless methods have been proposed for its solution (see, for example, [28] for a history, and the two volumes [24] and [25] for a detailed account of such methods). In the 1800's, it was observed by Frobenius that the roots of the polynomial are the eigenvalues of a certain matrix called the *companion matrix*, formed using the polynomial coefficients. A matrix whose eigenvalues are the roots of $p(x)$ is called a *linearization* of $p(x)$. Given a linearization of $p(x)$, the roots of the polynomial can thus be recovered by computing the eigenvalues of the matrix.

If the roots of the polynomial are found numerically, than the computed roots can be viewed as the exact roots of a perturbed polynomial $p(x) + \delta p(x)$ with coefficients $c_i + \delta c_i$, where the size of the vector $\delta c$ is called the *backward error* in the polynomial coefficients. The best possible bound on the backward error that a linearization method can have for general polynomials is

$$\|\delta c\| \lesssim \|c\| u, \tag{2}$$

or, in other words, that the relative normwise backward error in the polynomial coefficients is bounded by machine precision $u$ (see, for example, [26]). The backward error in the companion matrix method was revealed by the influential paper [19], which analyzed the relationship between perturbations in the companion matrix and perturbations in the polynomial coefficients. There, the authors proved that if the companion matrix $C$ is perturbed by the matrix $E$, then the matrix $C + E$ is a linearization of the polynomial with perturbed coefficients $c_i + \delta c_i$, and that the perturbation satisfies the normwise bound

$$\|\delta c\| \lesssim \|c\| \|E\| u. \tag{3}$$

If the eigenvalues are computed by a standard QR algorithm, which is known to be backward stable (see, for example, [35]), then the computed eigenvalues are the exact eigenvalues of $C + E$, where $\|E\| \lesssim \|C\| u$. Since $\|C\| \approx \|c\|$, it follows that the backward error in the polynomial coefficients is bounded by

$$\|\delta c\| \lesssim \|c\|^2 u. \tag{4}$$

Thus, as $\|c\|$ get larger, the relative backward error in the coefficients increases. The companion matrix method, at first glance, would appear then to have two drawbacks: it falls short of the optimal backward error bound (2), and it costs $O(n^3)$ operations as a result of using the QR algorthim.

The situation improved dramatically in 2007, when Bini, Eidelman, Gemignani, and Gohberg published a paper [10] describing a stable, $O(n^2)$ explicit QR method for companion matrices (around the same time, Chandrasekaran, Gu, Xia, and Zhu also discovered an $O(n^2)$ method for companion matrices, see [15]). The algorithm is based on the observation that the companion matrix and its QR iterates have a certain structure which allows them to be represented by a collection of $O(n)$ parameters called *generators* (specifically, the companion matrix is a Hessenberg matrix that is the sum of a unitary matrix and a rank-1 perturbation; matrices of this form are called *fellow matrices*). In 2010, an implicit version of this algorithm, also stable and $O(n^2)$, and also based on generators, was introduced in [9]. Around the same time, Van Barel, Vandebril, Van Dooren, and Frederix discovered in [8] an alternative stable, $O(n^2)$ implicit QR algorithm based on representing the unitary part by so-called *core transformations*, which are rotation matrices acting only on two adjacent rows at a time (see, for example, [2]). The first example of a proof of backward stability for an implicit $O(n^2)$ QR algorithm for companion matrices was given by Aurentz, Mach, Vandebril, and Watkins in [4]; this algorithm is again based on core transformations. The backward stability result accompanying this QR algorithm guarantees the sub-optimal bound (2), but has the optimal complexity of $O(n^2)$. Amazingly, the authors then discovered that the algorithm

they had constructed, with some minor modifications, actually yields the optimal bound (4) in practice. An investigation showed that the reason for this remarkable behavior is that their algorithm is not just backward stable, but is *componentwise* backward stable, meaning that the backward error in the companion matrix can be decomposed into proportional backward errors in each of its components. They published a proof of the componentwise backward stability of their algorithm, together with a proof that componentwise backward stability guarantees the bound (2), in [3], along with numerical experiments.

Thus, if the coefficients of the polynomial $p(x)$ in the monomial basis are known, then the algorithm of [3] is optimal in both error and time complexity. However, if the coefficients are not known, then the companion matrix cannot be used to the find the roots accurately, since the relationship between the values of the polynomial $p(x)$ and the coefficients of its monomial expansion is highly unstable (this fact has been known for many decades, at least as early as Wilkinson [31]). If the polynomial $p(x)$ is instead expanded in a basis of Chebyshev polynomials

$$p(x) = c_0 + c_1 T_1(x) + \cdots + c_{n-1} T_{n-1}(x) + T_n(x), \tag{5}$$

where $T_i(x)$ is the Chebyshev polynomial of order $i$, then the relationship between the coefficients and the polynomial is perfectly stable (see, for example, [36]). In fact, this observation is the basis for the Chebfun software package (see [6] and [16]). An analogue of the companion matrix, constructed from the Chebyshev expansion coefficients, was discovered in 1961 by Good [21], who called it the *colleague matrix*, and independently by Spect in 1957 [33]–[34]. The first $O(n^2)$ algorithm for colleague matrices was discovered by Bini, Gemignani, and Pan in 2005 (even before [10] appeared) and is a stable, explicit QR algorithm based on generators [11]. Like the companion matrix, the colleague matrix has a special structure that is preserved over QR iterations (specifically, the colleague matrix is a Hessenberg matrix that is the sum of a Hermitian matrix and a rank-1 perturbation). In 2008, Eidelman, Gemignani, and Gohberg, in [20], introduced a stable, $O(n^2)$ implicit QR algorithm. The relationship between the backward error in the Chebyshev expansion coefficients and perturbations to the colleague matrix was first investigated Nakasukasa and Noferini in [26], where the authors found a lower bound for the backward error in the coefficients, showing that a backward stable QR algorithm can do no better than (4) (around the same time, Lawrence, Van Barel, and Van Dooren published a general analysis in [23], where they also proved a lower bound for colleague matrices). In [30], Perez and Noferini improved on this result and found an upper bound as well, proving that if the perturbation to the colleague matrix is small, then the bound (2) is achieved. The relationship between componentwise perturbations to the colleague matrix and the backward error in the coefficients was described completely in 2019 by Noferini, Robol, and Vandebril in [27].

Recently (see [32] and [14]), it was observed that certain $O(n^2)$ structured QR algorithms for colleague matrices are surprisingly stable, attaining the bound (2) in many cases, an observation that mirrors the discovery in [3] for the case of companion matrices. However, unlike in [3], all previously proposed $O(n^2)$ structured QR algorithms for colleague matrices have polynomials for which the worst-case bound (4) is attained. Thus, the question of whether or not there exists a structured $O(n^2)$ QR algorithm that, when used to find the roots of a colleague matrix, attains the optimal bound (2), has

remained open. In this report, we answer this question in the affirmative by presenting a new, explicit $O(n^2)$ QR algorithm for colleague matrices (in fact, for all Hessenberg matrices that have a Hermitian plus rank-1 structure), and prove that our algorithm is componentwise backward stable. Combined with the result in [27], this amounts to a proof that the optimal bound (2) is attained for all polynomials $p(x)$. We demonstrate that this is indeed the case with several numerical experiments.

The structure of this report is as follows. Section 2 describes the mathematical and numerical preliminaries. Section 3 describes the algorithm, and explains the significance of each step. In Section 4, we prove rigorously that the algorithm is componentwise backward stable. Section 5 presents the results of several numerical experiments. In Section 6, we discuss possible extensions and generalizations of the algorithm.

# 2 Preliminaries

In this section, we describe the mathematical and numerical preliminaries.

## 2.1 Linear Algebra

The following lemma states that if the sum of a Hermitian matrix and a rank-1 update $pq^*$ is lower Hessenberg, then the matrix is determined entirely by its diagonal and superdiagonal together with the vectors $p$ and $q$.

**Lemma 2.1** (Eidelman, Gemignani, Gohberg [20])**.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is Hermitian, and let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that $p, q \in \mathbb{C}^n$ and that $A + pq^*$ is lower Hessenberg. Then*

$$a_{i,j} = \begin{cases} -p_i q_j^* & \text{if } j > i+1 \\ \beta_i & \text{if } j = i+1 \\ d_i & \text{if } j = i \\ \overline{\beta_j} & \text{if } j = i-1 \\ -q_j p_i^* & \text{if } j < i-1 \end{cases} \tag{6}$$

*where $a_{i,j}$ denotes the $(i,j)$-th entry of $A$.*

The following lemma states that if the sum of a matrix and a rank-1 update $pq^*$ is lower triangular, then the upper Hessenberg part of the matrix is determined entirely by its diagonal and subdiagonal, together with the vectors $p$ and $q$.

**Lemma 2.2.** *Suppose that $B \in \mathbb{C}^{n\times n}$ and let $d$ and $\gamma$ denote the diagonal and subdiagonal of $B$, respectively. Suppose that $p, q \in \mathbb{C}^n$ and that $B + pq^*$ is lower triangular. Then*

$$b_{i,j} = \begin{cases} -p_i q_j^* & \text{if } j > i \\ d_i & \text{if } j = i \\ \gamma_j & \text{if } j = i-1 \end{cases} \tag{7}$$

*where $b_{i,j}$ denotes the $(i,j)$-th entry of $B$.*

The following definition introduces two matrix seminorms that we will need in our error analysis.

**Definition 2.1.** Suppose that $A \in \mathbb{C}^{n\times n}$ and let $a_{i,j}$ denote the $(i,j)$-th entry of $A$. We will use the notation $\|\cdot\|_H$ to denote the square root of the sum of squares of the entries in the upper Hessenberg part of a matrix, so that

$$\|A\|_H = \sqrt{\sum_{j\geq i-1} |a_{i,j}|^2}. \tag{8}$$

Likewise, we will use the notation $\|\cdot\|_T$ to denote the square root of the sum of squares of the entries in the upper triangular part, so that

$$\|A\|_T = \sqrt{\sum_{j\geq i} |a_{i,j}|^2}. \tag{9}$$

The following is a straightforward lemma stating that if a certain sequence of transformations is applied to a matrix on the right, then the upper triangular part of the result is determined by only the upper Hessenberg part of the original matrix.

**Lemma 2.3.** *Suppose that $B \in \mathbb{C}^{n\times n}$, and let $P_2, P_3, \ldots, P_n \in \mathbb{C}^{n\times n}$ be matrices such that $P_k$ only affects the $(k-1,k)$-plane of any vector it is applied to. Define $P \in \mathbb{C}^{n\times n}$ by the formula $P = P_2 P_3 \cdots P_n$. Then the upper triangular part of $BP^*$ is determined entirely by the upper Hessenberg part of $B$. Furthermore, if $P_2, P_3, \ldots, P_n$ are unitary, then $\|BP^*\|_T \leq \|B\|_H$.*

## 2.2 Error Analysis

The following definition introduces the notation used in the error analysis that appears in this report. We follow the notation used in [22] and [3].

**Definition 2.2.** Evaluation of an expression in floating point arithmetic is denoted by $fl(\cdot)$, and we denote the unit roundoff (or machine epsilon) by $u$. We assume that

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \qquad |\delta| \leq u, \tag{10}$$

where op stands for any of the basic arithmetic operations $+, -, *, /$. We denote computed quantities by a hat, so that $\widehat{x}$ denotes the computed approximation to $x$. We use the notation $\lesssim$ to mean "less than or equal to the right hand side times a modest multiplicative constant depending on $n$ as a low-degree polynomial", where the meaning of $n$ is clear from the context. Whenever a matrix or vector norm appears to the left or right of $\lesssim$, we omit the particular choice of norm, since in finite dimensions all norms are equivalent. When $u$ appears in an expression on the right hand side of $\lesssim$, we ignore all higher order powers of $u$.

### 2.2.1 Floating point computation of complex plane rotations

The following lemma bounds the forward error of the floating point computation of a complex plane rotation (see, for example, §20 of [37]).

**Lemma 2.4.** *Suppose that $x = (x_1, x_2)^T \in \mathbb{C}^2$, and let $Q \in \mathrm{SU}(2)$ be the complex rotation matrix which eliminates the first entry, so that $(Qx)_1 = 0$. Let $\widehat{Q} \in \mathbb{C}^{2\times 2}$ be the floating point matrix defined by*

$$\widehat{Q}_{1,1} = c \qquad \widehat{Q}_{1,2} = -s \tag{11}$$

$$\widehat{Q}_{2,1} = \overline{s} \qquad \widehat{Q}_{2,2} = \overline{c} \tag{12}$$

$$\tag{13}$$

*where $c = fl\big(x_2/\sqrt{|x_1|^2 + |x_2|^2}\big)$ and $s = fl\big(x_1/\sqrt{|x_1|^2 + |x_2|^2}\big)$, and where $c = 1$ and $s = 0$ if $\|x\| = 0$. Then*

$$\|\widehat{Q} - Q\| \lesssim u. \tag{14}$$

### 2.2.2 Multiplication by complex plane rotations

The following lemma estimates the forward error of applying a plane rotation to a vector (see, for example, §21 of [37]).

**Lemma 2.5.** *Suppose that $Q \in \mathrm{SU}(2)$ is a complex rotation matrix, and let $\widehat{Q}$ be a floating point approximation to $Q$ satisfying (14). Suppose further that $x = (x_1, x_2)^T \in \mathbb{C}^2$. Then*

$$\|fl(\widehat{Q}x) - Qx\| \lesssim \|x\|u. \tag{15}$$

## 2.3 Colleague Matrices and Polynomial Rootfinding

Suppose that $p(x)$ is a monic polynomial of order $n$ represented by

$$p(x) = \sum_{j=0}^{n} c_j T_j(x), \tag{16}$$

where $c_j \in \mathbb{R}$, $c_n = 1$, and $T_j(x)$ is the Chebyshev polynomial of order $j$. It turns out that the roots of $p(x)$ are the eigenvalues of the (scaled) $n \times n$ colleague matrix

$$C = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & & & \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & \ddots & \ddots & \\ & & \ddots & 0 & \frac{1}{2} \\ & & & \frac{1}{2} & 0 \end{pmatrix} - \frac{1}{2} e_n \begin{pmatrix} c_0\sqrt{2} & c_1 & c_2 & \cdots & c_{n-1} \end{pmatrix}, \tag{17}$$

where $e_n$ is the $n$-th unit vector (see, for example, [21]). A matrix $C$ whose eigenvalues are the roots of $p(x)$ is called a *linearization* of $p(x)$. Letting

$$A = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & & & \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & \ddots & \ddots & \\ & & \ddots & 0 & \frac{1}{2} \\ & & & \frac{1}{2} & 0 \end{pmatrix} \tag{18}$$

and

$$q^* = -\frac{1}{2}\begin{pmatrix} c_0\sqrt{2} & c_1 & c_2 & \cdots & c_{n-1} \end{pmatrix}, \tag{19}$$

we see that the colleague matrix $C$ can be written as

$$C = A + e_n q^*, \tag{20}$$

where $A$ is Hermitian and $C$ is lower Hessenberg.

The following beautiful theorem by Noferini, Robol, and Vandebril (see Corollary 5.4 of [27]) bounds the change in the coefficients of the polynomial being linearized by the componentwise perturbations of the colleague matrix $C = A + e_n q^*$.

**Theorem 2.6.** *Let $C = A + e_n q^*$ be the linearization (17) of the monic polynomial $p(x)$, expressed in the Chebyshev basis. Consider the perturbations $\|\delta A\| \leq \epsilon_A$, $\|\delta e_n\| \leq \epsilon_n$, and $\|\delta q\| \leq \epsilon_q$. Then, the matrix*

$$C + \delta C = A + \delta A + (e_n + \delta e_n)(q + \delta q)^* \tag{21}$$

*is a linearization of the polynomial*

$$p(x) + \delta p(x) = \sum_{j=0}^{n} (c_j + \delta c_j) T_j(x), \tag{22}$$

*where $\|\delta c\| \lesssim \epsilon_n + \epsilon_q + \|c\|\epsilon_A$.*

## 2.4 Stability of Rootfinding Using Linearizations

Suppose that $p(x)$ is a monic polynomial of order $n$ represented in the Chebshev basis

$$p(x) = \sum_{j=0}^{n} c_j T_j(x), \tag{23}$$

where $c_j \in \mathbb{R}$, $c_n = 1$, and $T_j(x)$ is the Chebyshev polynomial of order $j$, and let the roots of $p(x)$ be denoted by $x_1, x_2, \ldots x_n \in \mathbb{C}$. Suppose that a rootfinding algorithm returns the computed roots $\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_n \in \mathbb{C}$. If the computed roots are the exact roots of some perturbed polynomial

$$p(x) + \delta p(x) = \sum_{j=0}^{n} (c_j + \delta c_j) T_j(x), \tag{24}$$

where

$$\frac{\|\delta c\|}{\|c\|} \lesssim u, \tag{25}$$

then we say that the rootfinding algorithm is backward stable. In fact, this is the best backward stability bound that can be hoped for, for general polynomials $p(x)$ (see the discussion in Appendix A of [26]).

**Remark 2.1.** Suppose that $p(x)$ is a polynomial of order $n$, that is *not* monic, expressed in the Chebyshev basis

$$p(x) = \sum_{j=0}^{n} a_j T_j(x), \tag{26}$$

where $a_j \in \mathbb{R}$ and $T_j(x)$ is the Chebyshev polynomial of order $j$. Clearly, the roots of $p(x)$ are identical to the roots of $p(x)/a_n$. Let $c_j = a_j/a_n$, for $j = 0, 1, \ldots, n$. If a backward stable rootfinding algorithm is applied to the monic polynomial $p(x)/a_n$, then, letting $\delta a = \delta c \cdot a_n$, the algorithm is also backward stable with respect to the original coefficients $a_j$, since

$$\frac{\|\delta a\|}{\|a\|} = \frac{\frac{1}{a_n}\|\delta a\|}{\frac{1}{a_n}\|a\|} = \frac{\|\delta c\|}{\|c\|} \lesssim u. \tag{27}$$

When linearization is used as a rootfinding algorithm, the stability of the computed roots comes from the stability of the eigenvalue algorithm applied to the colleague matrix $C$. If the eigenvalues of $C$ are computed by an unstructured QR algorithm, then the backward error $\delta C$ on $C$ is bounded by $\|\delta C\| \lesssim \|C\|u$. Since the backward error is unstructured, it follows that $\|\delta A\| \approx \|C\|u$, so, by Theorem 2.6 together with the fact that $\|C\| \approx \|c\|$, the backward error in the coefficients is bounded by $\|\delta c\| \lesssim \|c\|^2 u$.

**Remark 2.2.** This backward error can be reduced by partially balancing the matrix $C$. Suppose that, instead of computing the eigenvalues of $C$, we compute the eigenvalues of

$$\widetilde{C} = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & & & \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & & \\ & \frac{1}{2} & \ddots & \ddots & \\ & & \ddots & 0 & \frac{\|c\|^{\frac{1}{2}}}{2} \\ & & & \frac{1}{2\|c\|^{\frac{1}{2}}} & 0 \end{pmatrix} - \frac{1}{2\|c\|^{\frac{1}{2}}} e_n \begin{pmatrix} c_0\sqrt{2} & c_1 & c_2 & \cdots & \|c\|^{\frac{1}{2}} c_{n-1} \end{pmatrix}. \tag{28}$$

Provided that the entry in the $(n,n)$-position is small, we have that $\|\widetilde{C}\| \approx \|c\|^{\frac{1}{2}}$, so the backward error $\delta\widetilde{C}$ of unstructured QR is bounded by $\|\delta\widetilde{C}\| \lesssim \|c\|^{\frac{1}{2}} u$. In practice, it turns out that $\|\delta A\| \approx \|\delta\widetilde{C}\| \lesssim \|c\|^{\frac{1}{2}}$, so $\|\delta c\| \lesssim \|c\|^{\frac{3}{2}} u$. The assumption that the $(n,n)$-th element is small is not always satisfied. However, usually the norm of $c$ is large because the last coefficient $a_n$ in the non-monic expansion (26) is small. When this is the case, we can simply raise the order of the expansion by one by taking an additional term. The last two terms will both be small and roughly the same size, making the $(n,n)$-th element small. Notice also that, by adjusting the last row, the matrix $\widetilde{C}$ can be represented as a symmetric tridiagonal matrix of magnitude $\|c\|^{\frac{1}{2}}$ plus a rank-1 matrix of magnitude $\|c\|^{\frac{1}{2}}$.

**Remark 2.3.** Let $\texttt{bal}(C)$ denote the matrix $C$ after complete balancing. Remarkably, in some situations, $\|\texttt{bal}(C)\| \approx 1$ even when $\|c\|$ is large. Thus, complete balancing can completely eliminate large entries in $C$, at the expense of destroying its symmetric tridiagonal plus rank-1 structure. See Remark 5.1, as well as the paper [29], for a more detailed discussion.

**Remark 2.4.** While unstructured QR, applied to the colleague matrix, is known to achieve only the backward error bound $\|\delta c\| \lesssim \|c\|^2 u$, the QZ algorithm, applied to an appropriately scaled matrix pencil, does result in a backward stable rootfinder with the bound $\|\delta c\| \lesssim \|c\| u$ (see, for example, [26]). This is because the eigenvalue problem for the colleague matrix can be written as a matrix pencil $A - \lambda B$, where both $A$ and $B$ are small, and a backward stable QZ algorithm applied to the pencil computes the exact eigenvalues of a perturbed pencil $(A + \delta A) - \lambda(B + \delta B)$, where $\|\delta A\| \lesssim \|A\| u$ and $\|\delta B\| \lesssim \|B\| u$. Unfortunately, it appears to be very difficult to construct structured, $O(n^2)$ QZ algorithms for colleague matrices that retain the nice stability properties of the unstructured, $O(n^3)$ QZ algorithm.

The following theorem, stated in a slightly different form in [27], says that if the eigenvalues of the colleague matrix $C = A + e_n q^*$ are computed using a componentwise backward stable algorithm, then linearization is backward stable as a rootfinding algorithm. It follows immediately from Theorem 2.6.

**Theorem 2.7.** *Suppose that the eigenvalues of the colleague matrix $C = A + e_n q^*$ are computed by a componentwise backward stable algorithm, in the sense that the computed eigenvalues are the exact eigenvalues of the matrix*

$$C + \delta C = A + \delta A + (e_n + \delta e_n)(q + \delta q)^*, \tag{29}$$

*where $\|\delta A\| \lesssim \|A\| u \approx u$, $\|\delta e_n\| \lesssim \|e_n\| u \approx u$, and $\|\delta q\| \lesssim \|q\| u$. Then, linearization is backward stable as a rootfinding algorithm, with $\|\delta c\| \lesssim \|c\| u$.*

### 2.5 Conventions

It was pointed out to the authors that, while the Hessenberg matrices in this report are all lower Hessenberg, the standard convention in numerical linear algebra is to study the transpose of the problem, and consider only upper Hessenberg matrices (see [18]). The upper Hessenberg form is much better notationally since, in upper Hessenberg form, the first elimination step eliminates the entry in the $(2, 1)$-position, while, in lower Hessenberg form, the entry in the $(n - 1, n)$-position is eliminated first. Furthermore, the upper Hessenberg form is more convenient when representing polynomials in the Lagrange basis (see, for example, [17]). Unfortunately, at the time that this was all pointed out, most of the writing and numerical codes were complete, and had been written in lower Hessenberg form because of a historical fluke related to the structure of old explicit QR codes that were used as a template for our algorithm.

## 3 The Algorithm

In this section, we give an overview of our algorithm. We begin by describing the class of matrices our algorithm can be applied to. Let $\mathcal{F}_n \subset \mathbb{C}^{n \times n}$ be the set of lower Hessenberg matrices of the form

$$A + pq^*, \tag{30}$$

where $A \in C^{n \times n}$ is Hermitian and $p, q^* \in \mathbb{C}^n$. Eidelman, Gemignani, and Gohberg observed in [20] that the matrix $A$ is determined entirely by:

1. The diagonal entries $d_i = a_{i,i}$, for $i = 1, 2, \ldots, n$;

2. The superdiagonal entries $\beta_i = a_{i,i+1}$ for $i = 1, 2, \ldots, n-1$;

3. The vectors $p$ and $q$

(see Lemma 2.1). Following [20], we call these four vectors the *basic elements* or *generators* of $A$. In [20], the authors construct an implicit QR algorithm that takes advantage of this structure to achieve a cost of $O(n^2)$. They also prove that their algorithm is backward stable, in the sense that, when the algorithm is used to compute the eigenvalues of a matrix $C \in \mathcal{F}_n$, the computed eigenvalues are the exact eigenvalues of $C + \delta C$, where $\|\delta C\| \lesssim \|C\|u$. This is the same backward stability bound that is provided by an unstructured QR algorithm.

*In this report, we describe a new explicit QR algorithm for matrices $A + pq^* \in \mathcal{F}_n$, that also has the cost $O(n^2)$, and prove that our algorithm is* componentwise *backward stable, in the sense that the computed eigenvalues are the exact eigenvalues of $(A + \delta A) + (p + \delta p)(q + \delta q)^*$, where $\|\delta A\| \lesssim \|A\|u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$.*

To motivate our algorithm, consider first the naive unshifted QR algorithm in exact arithmetic, applied to the matrix $C = A + pq^*$. Let the matrix $U_n \in C^{n\times n}$ be the unitary matrix that rotates the $(n-1, n)$-plane so that

$$(U_n C)_{n-1,n} = 0, \tag{31}$$

eliminating the superdiagonal in the $(n-1, n)$-th position. Likewise, let $U_{n-1} \in C^{n\times n}$ denote the unitary matrix rotating the $(n-2, n-1)$-plane so that

$$(U_{n-1}U_n C)_{n-2,n-1} = 0, \tag{32}$$

eliminating the superdiagonal in the $(n-2, n-1)$-th position. Continuing in this fashion, let $U_{n-2}, U_{n-3}, \ldots, U_2$ be the unitary matrices eliminating the superdiagonal entries in the $(n-3, n-2), (n-4, n-3), \ldots, (1, 2)$ positions of the matrices $(U_{n-1}U_n C), (U_{n-2}U_{n-1}U_n C)$, $\ldots, (U_3 \cdots U_{n-1}U_n C)$, respectively. Letting $U = U_2 U_3 \cdots U_n$, we have that $UC$ is lower triangular. This matrix has the form

$$UC = B + (Up)q^*, \tag{33}$$

where the upper Hessenberg part of the matrix $B = UA$ is determined entirely by:

1. The diagonal entries $\underline{d}_i = b_{i,i}$, for $i = 1, 2, \ldots, n$;

2. The subdiagonal entries $\underline{\gamma}_i = b_{i+1,i}$, for $i = 1, 2, \ldots, n-1$;

3. The vectors $\underline{p} = Up$ and $q$

(see Lemma 2.2). Like with the matrix $A$, we call these four vectors the *basic elements* or *generators* of (the upper Hessenberg part of) $B$.

Next, the matrix is multiplied by $U^*$ on the right; clearly,

$$UCU^* = BU^* + Up(Uq)^*, \tag{34}$$

so

$$UCU^* = UAU^* + Up(Uq)^*. \tag{35}$$

It's easy to show that, since $UC$ is lower triangular and $U^* = U_n^* U_{n-1}^* \cdots U_2^*$, where $U_k$ rotates the $(k-1,k)$-plane, the matrix $UCU^*$ is lower Hessenberg. Thus, $UCU^* \in \mathcal{F}_n$, and the matrix $\underline{A} = UAU^*$ is determined entirely by its diagonal and superdiagonal, together with $\underline{p} = Up$ and $\underline{q} = Uq$. Furthermore, the upper triangular part of $UAU^*$ is determined entirely by the upper Hessenberg part of $B$ (see Lemma 2.3), and since $UAU^*$ is Hermitian, it follows that the whole of the matrix $UAU^*$ is determined entirely by the upper Hessenberg part of $B$.

In our algorithm, we use only the basic elements of $A$ and $B$ to represent our matrices. This results in a single iteration of our QR algorithm requiring $O(n)$ operations. Furthermore, we prove that the matrix $\widehat{\underline{A}}$ and vectors $\widehat{\underline{p}}$ and $\widehat{\underline{q}}$, computed by a single iteration of our QR algorithm, have the componentwise forward error bounds $\|\widehat{\underline{A}} - \underline{A}\| \lesssim \|A\|u$, $\|\widehat{\underline{p}} - \underline{p}\| \lesssim \|p\|u$, and $\|\widehat{\underline{q}} - \underline{q}\| \lesssim \|p\|u$. We then show that these componentwise forward error bounds result in componentwise backward stability.

### 3.1 Eliminating the Superdiagonal

In this section, we describe how our algorithm performs a single elimination of a superdiagonal element (see Algorithm 1). Suppose that we have already eliminated the superdiagonal elements in the positions $(n-1,n),(n-2,n-1),\ldots,(k,k+1)$. Let $p^{(k+1)} = U_{k+1}U_{k+2}\cdots U_n p$ and $B^{(k+1)} = U_{k+1}U_{k+2}\cdots U_n A$. Suppose further that $\widehat{p}^{(k+1)}$ and $\widehat{B}^{(k+1)}$ are the computed approximations to $p^{(k+1)}$ and $B^{(k+1)}$, and that the upper Hessenberg part of the computed matrix $\widehat{B}^{(k+1)}$ is represented by its generators:

1. The diagonal elements $\widehat{d}_i^{(k+1)} = \widehat{b}_{i,i}^{(k+1)}$, for $i = 1,2,\ldots,n$;
2. The superdiagional elements $\widehat{\beta}_i^{(k+1)} = \widehat{b}_{i,i+1}^{(k+1)}$, for $i = 1,2,\ldots,k-1$;
3. The subdiagional elements $\widehat{\gamma}_i^{(k+1)} = \widehat{b}_{i+1,i}^{(k+1)}$, for $i = 1,2,\ldots,n-1$;
4. The vectors $\widehat{p}^{(k+1)}$ and $q$, from which the remaining elements in the upper Hessenberg part are inferred.

Suppose that $\|\widehat{B}^{(k+1)} - B^{(k+1)}\|_H \lesssim \|A\|u$ and $\|\widehat{p}^{(k+1)} - p^{(k+1)}\| \lesssim \|p\|u$. Notice that, if we define $\widehat{B}^{(n+1)} = B^{(n+1)} = A$ and $\widehat{p}^{(n+1)} = p^{(n+1)} = p$, then this is obviously true for $k = n$.

To eliminate the superdiagonal element in the $(k-1,k)$ position of $\widehat{B}^{(k+1)} + \widehat{p}^{(k+1)}q^*$, we first compute the rotation matrix $Q_k \in \mathrm{SU}(2)$ that eliminates it by a rotation in the $(k-1,k)$-plane (see Line 4 of Algorithm 1). Next, we apply the rotation matrix separately to the generators of $\widehat{B}^{(k+1)}$ and to the vector $\widehat{p}^{(k+1)}$. Since we are only interested in computing the upper Hessenberg part of $\widehat{B}^{(k)}$, we need to update the subdiagonal element in the $(k-1,k-2)$ position of $\widehat{B}^{(k+1)}$, represented by $\widehat{\gamma}_{k-2}^{(k+1)}$ (see Figure 1). However, this calculation requires the sub-subdiagonal entry in the $(k,k-2)$ position of $\widehat{B}^{(k+1)}$, which is unknown to us since only the upper Hessenberg part of $\widehat{B}^{(k+1)}$ is available. Fortunately, it can be recovered by the following trick.

$$\left(\begin{array}{ccccccc}
\ddots & \widehat{\gamma}_{k-3}^{(k+1)} & \widehat{d}_{k-2}^{(k+1)} & \widehat{\beta}_{k-2}^{(k+1)} & -\widehat{p}_{k-2}^{(k+1)} q_k^* & -\widehat{p}_{k-2}^{(k+1)} q_{k+1}^* & \cdots \\
\hline
\cdots & \times & \widehat{\gamma}_{k-2}^{(k+1)} & \widehat{d}_{k-1}^{(k+1)} & \widehat{\beta}_{k-1}^{(k+1)} & -\widehat{p}_{k-1}^{(k+1)} q_{k+1}^* & \cdots \\
\cdots & \times & \widehat{b}_{k,k-2}^{(k+1)} & \widehat{\gamma}_{k-1}^{(k+1)} & \widehat{d}_k^{(k+1)} & -\widehat{p}_k^{(k+1)} q_{k+1}^* & \cdots \\
\hline
\cdots & \times & \times & \times & \widehat{\gamma}_k^{(k+1)} & \widehat{d}_{k+1}^{(k+1)} & \ddots
\end{array}\right)$$

Figure 1: The $(k-1)$-th and $k$-th rows of $\widehat{B}^{(k+1)}$, represented by its generators.

Since $B^{(k+1)} = U_{k+1} U_{k+2} \cdots U_n A$ and $A$ is Hermitian, it follows that $B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*$ is also Hermitian. Thus,

$$(B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)_{k,k-2} = \overline{(B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)}_{k-2,k}. \tag{36}$$

Furthermore, since right-multiplication by $U_j^*$ only affects columns $j$ and $j-1$ (see Figure 1), we have that right-multiplication by $U_n^* U_{n-1}^* \cdots U_{k+1}^*$ leaves $b_{k,k-2}^{(k+1)}$ unchanged. Therefore,

$$b_{k,k-2}^{(k+1)} = \overline{(B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)}_{k-2,k}. \tag{37}$$

We know that the entries in the $(k-2,k), (k-2,k+1), \ldots, (k-2,n)$ positions of $\widehat{B}^{(k+1)}$ are inferred from $\widehat{p}^{(k+1)}$ and $q$ by the formula

$$\widehat{b}_{k-2,\ell}^{(k+1)} = -\widehat{p}_{k-2}^{(k+1)} q_\ell^*, \tag{38}$$

for $\ell = k, k+1, \ldots, n$. Combining (37) and (38), we thus have that the sub-subdiagonal entry in the $(k,k-2)$ position of $\widehat{B}^{(k+1)}$ can be recovered by the formula

$$\widehat{b}_{k,k-2}^{(k+1)} = \overline{(-\widehat{p}_{k-2}^{(k+1)} q^* U_n^* U_{n-1}^* \cdots U_{k+1}^*)_k}. \tag{39}$$

Defining $\widetilde{q}^{(k+1)} = U_{k+1} U_{k+2} \cdots U_n q$, we have

$$\widehat{b}_{k,k-2}^{(k+1)} = -\widetilde{q}_k^{(k+1)} \widehat{p}_{k-2}^{(k+1)*}. \tag{40}$$

By computing the vector $\widehat{\widetilde{q}}^{(k+1)}$ (see Line 14 of Algorithm 1), we use this formula to recover the sub-subdiagonal element $\widehat{b}_{k,k-2}^{(k+1)}$.

Thus, the element in the $(k-1,k-2)$ position of $\widehat{B}^{(k+1)}$, represented by $\widehat{\gamma}_{k-2}^{(k+1)}$, is updated in Line 6 of Algorithm 1. Next, the elements in the $(k-1,k-1)$ and $(k,k-1)$ positions of $\widehat{B}^{(k+1)}$, represented by $\widehat{d}_{k-1}^{(k+1)}$ and $\widehat{\gamma}_{k-1}^{(k+1)}$, respectively, are updated in a straightforward way in Line 8. Finally, the elements in the $(k-1,k)$ and $(k,k)$ positions of $\widehat{B}^{(k+1)}$, represented by $\widehat{\beta}_{k-1}^{(k+1)}$ and $\widehat{d}_k^{(k+1)}$, respectively, are updated in Line 9, and the vector $\widehat{p}^{(k+1)}$ is rotated in Line 10.

Since we've eliminated the superdiagonal element in the $(k-1,k)$ position of $\widehat{B}^{(k+1)} + \widehat{p}^{(k+1)}q^*$, we have that the $(k-1,k)$ element of the matrix $\widehat{B}^{(k)}$ is inferred from $\widehat{p}^{(k)}$ and $q$ by the formula

$$\widehat{b}^{(k)}_{k-1,k} = -\widehat{p}^{(k)}_{k-1}q^*_k. \tag{41}$$

Now, we would like the upper Hessenberg part of $\widehat{B}^{(k)}$ to have a small componentwise error, so that $\|\widehat{B}^{(k)} - B^{(k)}\|_H \lesssim \|A\|u$. However, consider the following scenario. Suppose that the norm of $(\widehat{p}^{(k+1)}_{k-1}q^*_k, \widehat{p}^{(k+1)}_k q^*_k)^T$ is much larger than $\|A\|$. By Lemma 2.5, the error in $\widehat{p}^{(k)}_{k-1}q^*_k$ will be approximately $\left(\sqrt{|\widehat{p}^{(k+1)}_{k-1}q^*_k|^2 + |\widehat{p}^{(k+1)}_k q^*_k|^2}\right)u$, which will be much larger than $\|A\|u$. In this situation then, even if $\|\widehat{p}^{(k+1)} - p^{(k+1)}\| \lesssim \|p\|u$ and $\|\widehat{B}^{(k+1)} - B^{(k+1)}\|_H \lesssim \|A\|u$, we will *not* have $\|\widehat{B}^{(k)} - B^{(k)}\|_H \lesssim \|A\|u$. To remedy this, we must apply a correction to $\widehat{p}^{(k)}_{k-1}$. Recall that the rotation matrix $Q_k$ was defined to be the matrix eliminating the $(k-1,k)$-th entry of $\widehat{B}^{(k+1)} + \widehat{p}^{(k+1)}q^*$ in exact arithmetic. If we let $(\mathring{p}^{(k)}_{k-1}, \mathring{p}^{(k)}_k)^T$ denote the result of applying $Q_k$ to $(\widehat{p}^{(k+1)}_{k-1}, \widehat{p}^{(k+1)}_k)^T$ in exact arithmetic, and likewise let $(\mathring{\beta}^{(k)}_{k-1}, \mathring{d}^{(k)}_k)^T$ denote the result of applying $Q_k$ to $(\widehat{\beta}^{(k+1)}_{k-1}, \widehat{d}^{(k+1)}_k)^T$ in exact arithmetic, then, by the definition of $Q_k$, we have

$$\mathring{\beta}^{(k)}_{k-1} + \mathring{p}^{(k)}_{k-1}q^*_k = 0. \tag{42}$$

By Lemma 2.5, we have that

$$|\mathring{\beta}^{(k)}_{k-1} - \widehat{\beta}^{(k)}_{k-1}| \lesssim \left(\sqrt{|\widehat{\beta}^{(k+1)}_{k-1}|^2 + |\widehat{d}^{(k+1)}_k|^2}\right)u \tag{43}$$

and

$$|\mathring{p}^{(k)}_{k-1}q^*_k - \widehat{p}^{(k)}_{k-1}q^*_k| \lesssim \left(\sqrt{|\widehat{p}^{(k+1)}_{k-1}q^*_k|^2 + |\widehat{p}^{(k+1)}_k q^*_k|^2}\right)u. \tag{44}$$

Thus, if $|\widehat{p}^{(k+1)}_{k-1}q^*_k|^2 + |\widehat{p}^{(k+1)}_k q^*_k|^2 > |\widehat{\beta}^{(k+1)}_{k-1}|^2 + |\widehat{d}^{(k+1)}_k|^2$, then we set

$$\widehat{p}^{(k)}_{k-1}q^*_k = -\widehat{\beta}^{(k)}_{k-1}, \tag{45}$$

so

$$\widehat{p}^{(k)}_{k-1} = -\widehat{\beta}^{(k)}_{k-1}/q^*_k \tag{46}$$

(see Line 12 of Algorithm 1). With this correction to $\widehat{p}^{(k)}_{k-1}$, it is easy to see that $\|\widehat{B}^{(k)} - B^{(k)}\|_H \lesssim \|A\|u$ and $\|\widehat{p}^{(k)} - p^{(k)}\| \lesssim \|p\|u$. If, on the other hand, $|\widehat{p}^{(k+1)}_{k-1}q^*_k|^2 + |\widehat{p}^{(k+1)}_k q^*_k|^2 \leq |\widehat{\beta}^{(k+1)}_{k-1}|^2 + |\widehat{d}^{(k+1)}_k|^2$, then the correction is not neccessary, since in this case the error in $\widehat{p}^{(k)}_{k-1}q^*_k$ is smaller than the error in $\widehat{\beta}^{(k)}_{k-1}$.

This process of eliminating the superdiagonal elements can be repeated, until the upper Hessenberg part of the matrix $\widehat{B}$, approximating $B = U_2U_3\cdots U_nA$, is obtained, together with $\widehat{\underline{p}}$, approximating $\underline{p} = U_2U_3\cdots U_np$ (see Algorithm 1). In Section 4.1, Lemma 4.1, we prove that the forward errors in the upper Hessenberg part of $\widehat{B}$ and in the vector $\widehat{\underline{p}}$ are proportional to $\|A\|u$ and $\|p\|u$, respectively.

**Algorithm 1** (A single elimination of the superdiagonal) **Inputs:** This algorithm accepts as inputs two vectors $d$ and $\beta$ representing the diagonal and superdiagonal, respectively, of an $n \times n$ Hermitian matrix $A$, as well as two vectors $p$ and $q$ of length $n$, where $A + pq^*$ is lower Hessenberg. **Outputs:** It returns as its outputs the rotation matrices $Q_2, Q_3, \ldots, Q_n \in \mathbb{C}^{2\times 2}$ so that, letting $U_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $Q_k$, $U_2 U_3 \cdots U_n (A + pq^*)$ is lower triangular. It also returns the vectors $\underline{d}$, $\underline{\gamma}$, and $\underline{p}$, where $\underline{d}$ and $\underline{\gamma}$ represent the diagonal and subdiagonal, respectively, of the matrix $U_2 U_3 \cdots U_n A$, and $\underline{p} = U_2 U_3 \cdots U_n p$.

1: Set $\gamma \leftarrow \overline{\beta}$, where $\gamma$ represents the subdiagonal.

2: Make a copy of $q$, setting $\widetilde{q} \leftarrow q$.

3: **for** $k = n, n-1, \ldots, 2$ **do**

4: Construct the $2 \times 2$ rotation matrix $Q_k \in \mathrm{SU}(2)$ so that

$$\left( Q_k \begin{bmatrix} \beta_{k-1} + p_{k-1} q_k^* \\ d_k + p_k q_k^* \end{bmatrix} \right)_1 = 0.$$

5: **if** $k \neq 2$ **then**

6: Rotate the subdiagonal and the sub-subdiagonal:

$$\gamma_{k-2} \leftarrow \left( Q_k \begin{bmatrix} \gamma_{k-2} \\ -\widetilde{q}_k p_{k-2}^* \end{bmatrix} \right)_1$$

7: **end if**

8: Rotate the diagonal and the subdiagonal: $\begin{bmatrix} d_{k-1} \\ \gamma_{k-1} \end{bmatrix} \leftarrow Q_k \begin{bmatrix} d_{k-1} \\ \gamma_{k-1} \end{bmatrix}$.

9: Rotate the superdiagonal and the diagonal: $\begin{bmatrix} \beta_{k-1} \\ d_k \end{bmatrix} \leftarrow Q_k \begin{bmatrix} \beta_{k-1} \\ d_k \end{bmatrix}$.

10: Rotate $p$: $\begin{bmatrix} p_{k-1} \\ p_k \end{bmatrix} \leftarrow Q_k \begin{bmatrix} p_{k-1} \\ p_k \end{bmatrix}$

11: **if** $|p_{k-1} q_k^*|^2 + |p_k q_k^*|^2 > |\beta_{k-1}|^2 + |d_k|^2$ **then**

12: Correct the vector $p$, setting $p_{k-1} \leftarrow -\frac{\beta_{k-1}}{q_k^*}$

13: **end if**

14: Rotate $\widetilde{q}$: $\begin{bmatrix} \widetilde{q}_{k-1} \\ \widetilde{q}_k \end{bmatrix} \leftarrow Q_k \begin{bmatrix} \widetilde{q}_{k-1} \\ \widetilde{q}_k \end{bmatrix}$

15: **end for**

16: Set $\underline{d} \leftarrow d$, $\underline{\gamma} \leftarrow \gamma$, and $\underline{p} \leftarrow p$.

$$\left(\begin{array}{c|c|c|c} \ddots & \vdots & \vdots & \vdots \\ \widehat{d}_{k-2}^{(k+1)} & -p_{k-2}\widehat{q}_{k-1}^{(k+1)*} & -p_{k-2}\widehat{q}_{k}^{(k+1)*} & -p_{k-2}\widehat{q}_{k+1}^{(k+1)*} \\ \gamma_{k-2} & \widehat{d}_{k-1}^{(k+1)} & -p_{k-1}\widehat{q}_{k}^{(k+1)*} & -p_{k-1}\widehat{q}_{k+1}^{(k+1)*} \\ \times & \gamma_{k-1} & \widehat{d}_{k}^{(k+1)} & \widehat{\beta}_{k}^{(k+1)} \\ \times & \times & \times & \widehat{d}_{k+1}^{(k+1)} \\ \vdots & \vdots & \vdots & \ddots \end{array}\right)$$

Figure 2: The $(k-1)$-th and $k$-th columns of $\widehat{A}^{(k+1)}$, represented by its generators.

### 3.2 Rotating Back to Hessenberg Form

In this section, we describe how our algorithm rotates the triangular matrix produced by an elimination of the superdiagonal back to lower Hessenberg form (see Algorithm 2). Suppose that $B$ is an $n \times n$ matrix and that $p$ and $q$ are vectors such that $B + pq^*$ is lower triangular. Notice that this condition is satisfied by the matrix $\widehat{B}$ and the vectors $\widehat{p}$ and $q$ from the preceding section, produced by an elimination of the superdiagonal. Let $\gamma$ denote the subdiagonal of $B$. Suppose that we have already applied the rotation matrices $U_n^*, U_{n-1}^*, \cdots, U_{k+1}^*$ to the right of $B$ and $q^*$, and let $q^{(k+1)} = U_{k+1}U_{k+2}\cdots U_n q$ and $A^{(k+1)} = BU_n^*U_{n-1}^*\cdots U_{k+1}^*$. Suppose that $\widehat{q}^{(k+1)}$ and $\widehat{A}^{(k+1)}$ are the computed approximations to $q^{(k+1)}$ and $A^{(k+1)}$, respectively, and that the upper triangular part of $\widehat{A}^{(k+1)}$ is represented by its generators:

1. The diagonal entries $\widehat{d}_i^{(k+1)} = \widehat{a}_{i,i}^{(k+1)}$, for $i = 1, 2, \ldots, n$;
2. The superdiagonal entries $\widehat{\beta}_i^{(k+1)} = \widehat{a}_{i,i+1}^{(k+1)}$ for $i = k, k+1, \ldots, n-1$;
3. The vectors $p$ and $\widehat{q}^{(k+1)}$, from which the remaining elements in the upper triangular part are inferred.

Suppose that $\|\widehat{A}^{(k+1)} - A^{(k+1)}\|_T \lesssim \|B\|_H u$ and $\|\widehat{q}^{(k+1)} - q^{(k+1)}\| \lesssim \|q\| u$. Notice that, if we define $\widehat{A}^{(n+1)} = A^{(n+1)} = B$ and $\widehat{q}^{(n+1)} = q^{(n+1)} = q$, then this is obviously true for $k = n$.

To apply the matrix $U_k^*$ to $\widehat{A}^{(k+1)} + p\widehat{q}^{(k+1)}$ on the right, we apply the matrix $Q_k^* \in \mathrm{SU}(2)$ separately to the generators of $\widehat{A}^{(k+1)}$ and to the vector $\widehat{q}^{(k+1)}$. We start by rotating the diagonal and superdiagonal elements in the $(k-1, k-1)$ and $(k-1, k)$ positions of $\widehat{A}^{(k+1)}$, represented by $\widehat{d}_{k-1}^{(k+1)}$ and $-p_{k-1}\widehat{q}_k^{(k+1)*}$, respectively, in Line 2 of Algorithm 2, saving the superdiagonal element in $\widehat{\beta}_{k-1}^{(k)}$ (see Figure 2). Next, we rotate the elements in the $(k, k-1)$ and $(k, k)$ positions, represented by $\gamma_{k-1}$ (the $(k-1)$-st element of the subdiagonal of $B$) and $\widehat{d}_k^{(k+1)}$, respectively, in a straightforward way in Line 3; since we are only interested in computing the upper triangular part of $\widehat{A}^{(k)}$, we only update the diagonal entry. Finally, we rotate the vector $\widehat{q}^{(k+1)}$ in Line 4.

The process of applying the rotation matrices on the right can be repeated, until the upper triangular part of the matrix $\widehat{\underline{A}}$, approximating $\underline{A} = BU_n^*U_{n-1}^* \cdots U_2^*$, is obtained, together with $\widehat{\underline{q}}$, approximating $\underline{q} = U_2U_3 \cdots U_nq$ (see Algorithm 2). In Section 4.1, Lemma 4.2, we prove that the forward errors in the upper triangular part of $\widehat{\underline{A}}$ and in the vector $\widehat{\underline{q}}$ are proportional to $\|B\|_H u$ and $\|q\|u$, respectively.

### 3.3 The QR Algorithms

The elimination of the superdiagonal described in Algorithm 1, followed by the rotation back to Hessenberg form described in Algorithm 2, can be iterated to find the eigenvalues of $A + pq^*$. Our unshifted explicit QR algorithm, based on this iteration, is described in Algorithm 3. This unshifted QR algorithm can be accelerated by the introduction of shifts; our explicit shifted QR algorithm, with Wilkinson shifts, is described in Algorithm 4.

In Section 4.1, we show that the forward error of one iteration of our QR algorithm (Algorithm 1 followed by Algorithm 2) satisfies componentwise forward error bounds. In Section 4.2, we use this result to prove that both our explicit unshifted QR algorithm (Algorithm 3) and our shifted QR algorithm (Algorithm 4) are componentwise backward stable.

---

**Algorithm 2** (Rotating the matrix back to Hessenberg form) **Inputs:** This algorithm accepts as inputs $n-1$ rotation matrices $Q_2, Q_2, \ldots, Q_n \in \mathbb{C}^{n\times n}$, two vectors $d$ and $\gamma$ representing the diagonal and subdiagonal, respectively, of an $n \times n$ complex matrix $B$, and two vectors $p$ and $q$ of length $n$, where $B + pq^*$ is lower triangular. **Outputs:** Letting $U_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $Q_k$, this algorithm returns as its outputs the vectors $\underline{d}$, $\underline{\beta}$, and $\underline{q}$, where $\underline{d}$ and $\underline{\beta}$ represent the diagonal and superdiagonal, respectively, of the matrix $BU_n^*U_{n-1}^* \cdots U_2^*$, and $\underline{q} = U_2U_3 \cdots U_nq$.

---

1: **for** $k = n, n-1, \ldots, 2$ **do**

2: Rotate the diagonal and the superdiagonal:

$$\begin{bmatrix} d_{k-1} \\ \beta_{k-1} \end{bmatrix} \leftarrow \overline{Q_k} \begin{bmatrix} d_{k-1} \\ -p_{k-1}q_k^* \end{bmatrix}.$$

3: Rotate the subdiagonal and the diagonal:

$$d_k \leftarrow \left(\overline{Q_k} \begin{bmatrix} \gamma_{k-1} \\ d_k \end{bmatrix}\right)_2$$

4: Rotate $q$: $\begin{bmatrix} q_{k-1} \\ q_k \end{bmatrix} \leftarrow Q_k \begin{bmatrix} q_{k-1} \\ q_k \end{bmatrix}$

5: **end for**

6: Set $\underline{d} \leftarrow d$, $\underline{\beta} \leftarrow \beta$, and $\underline{q} \leftarrow q$.

---

**Algorithm 3** (Unshifted explicit QR) **Inputs:** This algorithm accepts as inputs two vectors $d$ and $\beta$ representing the diagonal and superdiagonal, respectively, of an $n \times n$ Hermitian matrix $A$, as well as two vectors $p$ and $q$ of length $n$, where $A + pq^*$ is lower Hessenberg. It also accepts a tolerance $\epsilon > 0$, which determines the accuracy the eigenvalues are computed to. **Outputs:** It returns as its output the vector $\lambda$ of length $n$ containing the eigenvalues of the matrix $A + pq^*$.

1: **for** $i = 1, 2, \ldots, n-1$ **do**

2: **while** $\beta_i + p_i q_{i+1}^* \geq \epsilon$ **do** ▷ Check if $(A + pq^*)_{i,i+1}$ is close to zero

3: Perform one iteration of QR (one step of Algorithm 1 followed by one step of Algorithm 2) on the submatrix $(A + pq^*)_{i:n,i:n}$ defined by the vectors $d_{i:n}$, $\beta_{i:n-1}$, $p_{i:n}$, and $q_{i:n}$.

4: **end while**

5: **end for**

6: Set $\lambda \leftarrow d$.

**Algorithm 4** (Shifted explicit QR) **Inputs:** This algorithm accepts as inputs two vectors $d$ and $\beta$ representing the diagonal and superdiagonal, respectively, of an $n \times n$ Hermitian matrix $A$, as well as two vectors $p$ and $q$ of length $n$, where $A + pq^*$ is lower Hessenberg. It also accepts a tolerance $\epsilon > 0$, which determines the accuracy the eigenvalues are computed to. **Outputs:** It returns as its output the vector $\lambda$ of length $n$ containing the eigenvalues of the matrix $A + pq^*$.

1: **for** $i = 1, 2, \ldots, n-1$ **do**

2: Set $\mu_{\text{sum}} \leftarrow 0$.

3: **while** $\beta_i + p_i q_{i+1}^* \geq \epsilon$ **do** ▷ Check if $(A + pq^*)_{i,i+1}$ is close to zero

4: Compute the eigenvalues $\mu_1$ and $\mu_2$ of the $2 \times 2$ submatrix $\begin{bmatrix} d_i + p_i q_i^* & \beta_i + p_i q_{i+1}^* \\ \overline{\beta}_i + p_{i+1} q_i^* & d_{i+1} + p_{i+1} q_{i+1}^* \end{bmatrix}$. ▷ This is just $(A + pq^*)_{i:i+1,i:i+1}$

5: Set $\mu$ to whichever of $\mu_1$ and $\mu_2$ is closest to $d_i + p_i q_i^*$.

6: Set $\mu_{\text{sum}} \leftarrow \mu_{\text{sum}} + \mu$.

7: Set $d_{i:n} \leftarrow d_{i:n} - \mu$.

8: Perform one iteration of QR (one step of Algorithm 1 followed by one step of Algorithm 2) on the submatrix $(A + pq^*)_{i:n,i:n}$ defined by the vectors $d_{i:n}$, $\beta_{i:n-1}$, $p_{i:n}$, and $q_{i:n}$.

9: **end while**

10: Set $d_{i:n} \leftarrow d_{i:n} + \mu_{\text{sum}}$.

11: **end for**

12: Set $\lambda_i \leftarrow d_i + p_i q_i^*$, for $i = 1, 2, \ldots, n$.

# 4 Componentwise Backward Stability

The principal results of this section are Theorems 4.6 and 4.7, which state that our unshifted and shifted QR algorithms, respectively, are componentwise backward stable. In Section 4.1, we prove that the forward error of a single sweep of our unshifted QR algorithm satisfies componentwise bounds. In Section 4.2, we use these bounds show componentwise backward stability of our QR algorithms.

## 4.1 Forward Error Analysis of a Single Sweep of $QR$

Suppose that $A$ is Hermitian and $A + pq^*$ is lower Hessenberg. In this section, we prove in Theorem 4.3 that the forward errors in $A$, $p$, and $q$ of single sweep of our explicit $QR$ algorithm are proportional to $\|A\|u$, $\|p\|u$, and $\|q\|u$, respectively.

The following lemma bounds the forward error of Algorithm 1 (the elimination of the superdiagonal).

**Lemma 4.1.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is a Hermitian matrix, and that $p, q \in \mathbb{C}^n$. Suppose further that $A+pq^*$ is lower Hessenberg, and let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that Algorithm 1 is carried out in floating point arithmetic with $d$, $\beta$, $p$, and $q$ as inputs, and let $Q_2, Q_3, \ldots, Q_n \in \mathrm{SU}(2)$ be the unitary matrices generated by an exact step of Line 4 of Algorithm 1 applied to the computed vectors at that step. Let $U_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $Q_k$, and define $U \in \mathbb{C}^{n\times n}$ by the formula $U = U_2U_3\cdots U_n$. Suppose finally that $\underline{\widehat{d}}$, $\underline{\widehat{\gamma}}$, and $\underline{\widehat{p}}$ are the outputs generated by Algorithm 1, and define the upper Hessenberg part of the matrix $\widehat{B} \in \mathbb{C}^{n\times n}$ by the formula*

$$\widehat{b}_{i,j} = \begin{cases} -\underline{\widehat{p}}_i q_j^* & \text{if } j > i, \\ \underline{\widehat{d}}_i & \text{if } j = i, \\ \underline{\widehat{\gamma}}_j & \text{if } j = i-1. \end{cases} \tag{47}$$

*where $\widehat{b}_{i,j}$ denotes the $(i,j)$-th entry of $\widehat{B}$. Let $B = UA$ and $\underline{p} = Up$. Then*

$$\|\widehat{B} - B\|_H \lesssim \|A\|u \tag{48}$$

*and*

$$\|\underline{\widehat{p}} - \underline{p}\| \lesssim \|p\|u, \tag{49}$$

*where $\|\cdot\|_H$ denotes the square root of the sum of squares of the entries in the upper Hessenberg part of its argument (see Definition 2.1).*

**Proof.** Suppose that $\widehat{d}^{(k)}$, $\widehat{\gamma}^{(k)}$, $\widehat{\beta}^{(k)}$, $\widehat{p}^{(k)}$, and $\widehat{\widetilde{q}}^{(k)}$ denote the computed vectors in Algorithm 1 after the elimination of the superdiagonal elements in the positions $(n-1, n), (n-2, n-1), \ldots, (k-1, k)$. Suppose further that the upper Hessenberg part of the matrix $\widehat{B}^{(k)} \in \mathbb{C}^{n\times n}$ is defined by the formula

$$\widehat{b}_{i,j}^{(k)} = \begin{cases} -\widehat{p}_i^{(k)} q_j^* & \text{if } j > i+1 \text{ or if } j = i+1 \text{ and } j \geq k, \\ \widehat{\beta}_i^{(k)} & \text{if } j = i+1 \text{ and } j < k, \\ \widehat{d}_i^{(k)} & \text{if } j = i, \\ \widehat{\gamma}_i^{(k)} & \text{if } j = i-1, \end{cases} \tag{50}$$

where $\widehat{b}_{i,j}^{(k)}$ denotes the $(i,j)$-th entry of $\widehat{B}^{(k)}$. Clearly, $\underline{\widehat{d}} = \widehat{d}^{(2)}$, $\underline{\widehat{\gamma}} = \widehat{\gamma}^{(2)}$, $\underline{\widehat{p}} = \widehat{p}^{(2)}$, and $\widehat{B} = \widehat{B}^{(2)}$. Let $B^{(k)} = U_k U_{k+1} \cdots U_n A$ and $p^{(k)} = U_k U_{k+1} \cdots U_n p$. We will prove that $\|\widehat{B}^{(k)} - B^{(k)}\|_H \lesssim \|A\| u$ and $\|\widehat{p}^{(k)} - p^{(k)}\| \lesssim \|p\| u$, for each $k = n, n-1, \ldots, 2$.

We begin by proving this statement for $k = n$. From Line 4, we have that the matrix $Q_n \in \mathrm{SU}(2)$ satisfies

$$\left( Q_n \begin{bmatrix} \beta_{n-1} + p_{n-1} q_n^* \\ d_n + p_n q_n^* \end{bmatrix} \right)_1 = 0, \tag{51}$$

with the computed matrix $\widehat{Q}_n$ satisfying $\|\widehat{Q}_n - Q_n\| \lesssim u$ by Lemma 2.4. In Line 6, we have

$$\widehat{\gamma}_{n-2}^{(n)} = \mathit{fl}\left( \widehat{Q}_n \begin{bmatrix} \gamma_{n-2} \\ -\widetilde{q}_n p_{n-2}^* \end{bmatrix} \right)_1. \tag{52}$$

At this stage $\widetilde{q}$ is still equal to $q$ and, according to Lemma 2.1, $a_{n,n-2} = -q_n p_{n-2}^*$. By definition, $a_{n-1,n-2} = \gamma_{n-2}$. Therefore, by Lemma 2.5, we have that $|\widehat{\gamma}_{n-2}^{(n)} - b_{n-1,n-2}^{(n)}| \lesssim \|A\| u$, where $b_{i,j}^{(n)}$ denotes the $(i,j)$-th entry of $B^{(n)}$. In Line 8, we have

$$\begin{bmatrix} \widehat{d}_{n-1}^{(n)} \\ \widehat{\gamma}_{n-1}^{(n)} \end{bmatrix} = \mathit{fl}\left( \widehat{Q}_n \begin{bmatrix} d_{n-1} \\ \gamma_{n-1} \end{bmatrix} \right). \tag{53}$$

Since $a_{n-1,n-1} = d_{n-1}$ and $a_{n,n-1} = \gamma_{n-1}$, by Lemma 2.5, we have that $|\widehat{d}_{n-1}^{(n)} - b_{n-1,n-1}^{(n)}| \lesssim \|A\| u$ and $|\widehat{\gamma}_{n-1}^{(n)} - b_{n,n-1}^{(n)}| \lesssim \|A\| u$. In Line 9, we have

$$\begin{bmatrix} \widehat{\beta}_{n-1}^{(n)} \\ \widehat{d}_n^{(n)} \end{bmatrix} = \mathit{fl}\left( \widehat{Q}_n \begin{bmatrix} \beta_{n-1} \\ d_n \end{bmatrix} \right). \tag{54}$$

Since, by definition, $a_{n-1,n} = \beta_{n-1}$ and $a_{n,n} = d_n$, it follows from Lemma 2.5 that $|\widehat{\beta}_{n-1}^{(n)} - b_{n-1,n}^{(n)}| \lesssim \left( \sqrt{|\beta_{n-1}|^2 + |d_n|^2} \right) u \leq \|A\| u$ and $|\widehat{d}_n^{(n)} - b_{n,n}^{(n)}| \lesssim \left( \sqrt{|\beta_{n-1}|^2 + |d_n|^2} \right) u \leq \|A\| u$. In Line 10, we have

$$\begin{bmatrix} \widehat{p}_{n-1}^{(n)\dagger} \\ \widehat{p}_n^{(n)} \end{bmatrix} = \mathit{fl}\left( \widehat{Q}_n \begin{bmatrix} p_{n-1} \\ p_n \end{bmatrix} \right), \tag{55}$$

where $\widehat{p}_{n-1}^{(n)\dagger}$ is a temporary value that will be corrected later. Once again, Lemma 2.5 tells us that $|\widehat{p}_{n-1}^{(n)\dagger} - p_{n-1}^{(n)}| \lesssim \left( \sqrt{|p_{n-1}|^2 + |p_n|^2} \right) u \leq \|p\| u$ and $|\widehat{p}_n^{(n)} - p_n^{(n)}| \lesssim \left( \sqrt{|p_{n-1}|^2 + |p_n|^2} \right) u \leq \|p\| u$. Next, we observe that, by the definition of $Q_n$, we have that

$$b_{n-1,n}^{(n)} + p_{n-1}^{(n)} q_n^* = 0. \tag{56}$$

In Line 12, we apply a correction to $\widehat{p}_{n-1}^{(n)\dagger}$, so that

$$\widehat{p}_{n-1}^{(n)} = \begin{cases} -\widehat{\beta}_{n-1}^{(n)} / q_n^* & \text{if } |p_{n-1} q_n^*|^2 + |p_n q_n^*|^2 > |\beta_{n-1}|^2 + |d_n|^2, \\ \widehat{p}_{n-1}^{(n)\dagger} & \text{if } |p_{n-1} q_n^*|^2 + |p_n q_n^*|^2 \leq |\beta_{n-1}|^2 + |d_n|^2. \end{cases} \tag{57}$$

From this, we see that, if $|p_{n-1}q_n^*|^2 + |p_n q_n^*|^2 > |\beta_{n-1}|^2 + |d_n|^2$, then

$$\begin{aligned}
&|\widehat{p}_{n-1}^{(n)} - p_{n-1}^{(n)}| = |-\widehat{\beta}_{n-1}^{(n)}/q_n^* - p_{n-1}^{(n)}| \\
&= |b_{n-1,n}^{(n)}/q_n^* - \widehat{\beta}_{n-1}^{(n)}/q_n^*| \\
&= \frac{1}{|q_n^*|}|b_{n-1,n}^{(n)} - \widehat{\beta}_{n-1}^{(n)}| \\
&\lesssim \frac{\sqrt{|\beta_{n-1}|^2 + |d_n|^2}}{|q_n^*|}u \\
&\leq \Big(\sqrt{|p_{n-1}|^2 + |p_n|^2}\Big)u \\
&\leq \|p\|u. \qquad (58)
\end{aligned}$$

where the second equality is due to (56). Furthermore,

$$\begin{aligned}
&|-\widehat{p}_{n-1}^{(n)} q_n^* - b_{n-1,n}^{(n)}| = |\widehat{\beta}_{n-1}^{(n)} - b_{n-1,n}^{(n)}| \\
&\lesssim \Big(\sqrt{|\beta_{n-1}|^2 + |d_n|^2}\Big)u \\
&\leq \|A\|u. \qquad (59)
\end{aligned}$$

If, on the other hand, $|p_{n-1}q_n^*|^2 + |p_n q_n^*|^2 \leq |\beta_{n-1}|^2 + |d_n|^2$, then

$$\begin{aligned}
&|\widehat{p}_{n-1}^{(n)} - p_{n-1}^{(n)}| = |\widehat{p}_{n-1}^{(n)\dagger} - p_{n-1}^{(n)}| \\
&\lesssim \sqrt{|p_{n-1}|^2 + |p_n|^2}u \\
&\leq \|p\|u. \qquad (60)
\end{aligned}$$

Moreover,

$$\begin{aligned}
&|-\widehat{p}_{n-1}^{(n)} q_n^* - b_{n-1,n}^{(n)}| = |-\widehat{p}_{n-1}^{(n)\dagger} q_n^* - b_{n-1,n}^{(n)}| \\
&= |-\widehat{p}_{n-1}^{(n)\dagger} q_n^* + p_{n-1}^{(n)} q_n^*| \\
&\lesssim |q_n^*|\Big(\sqrt{|p_{n-1}|^2 + |p_n|^2}\Big)u \\
&\leq \Big(\sqrt{|\beta_{n-1}|^2 + |d_n|^2}\Big)u \\
&\leq \|A\|u. \qquad (61)
\end{aligned}$$

where the second equality follows from (56). This completes the proof that $\|\widehat{B}^{(n)} - B^{(n)}\|_H \lesssim \|A\|u$ and $\|\widehat{p}^{(n)} - p^{(n)}\| \lesssim \|p\|u$.

Now, we will show that, if $\|\widehat{B}^{(k+1)} - B^{(k+1)}\|_H \lesssim \|A\|u$ and $\|\widehat{p}^{(k+1)} - p^{(k+1)}\| \lesssim \|p\|u$, then $\|\widehat{B}^{(k)} - B^{(k)}\|_H \lesssim \|A\|u$ and $\|\widehat{p}^{(k)} - p^{(k)}\| \lesssim \|p\|u$. From Line 4, we have that the matrix $Q_k \in \mathrm{SU}(2)$ satisfies

$$\left(Q_k \begin{bmatrix} \widehat{\beta}_{k-1}^{(k+1)} + \widehat{p}_{k-1}^{(k+1)} q_k^* \\ \widehat{d}_k^{(k+1)} + \widehat{p}_k^{(k+1)} q_k^* \end{bmatrix}\right)_1 = 0, \qquad (62)$$

with the computed matrix $\widehat{Q}_k$ satisfying $\|\widehat{Q}_k - Q_k\| \lesssim u$ by Lemma 2.4. In Line 6, we have

$$\widehat{\gamma}_{k-2}^{(k)} = \mathit{fl}\left(\widehat{Q}_k \begin{bmatrix} \widehat{\gamma}_{k-2}^{(k+1)} \\ -\widehat{\widetilde{q}}_k^{(k+1)} \widehat{p}_{k-2}^{(k+1)*} \end{bmatrix}\right)_1. \tag{63}$$

We must first show that

$$\left|-\widehat{\widetilde{q}}_k^{(k+1)} \widehat{p}_{k-2}^{(k+1)*} - b_{k,k-2}^{(k+1)}\right| \lesssim \|A\| u. \tag{64}$$

We begin by observing that

$$b_{k,k-2}^{(k+1)} = (B^{(k+1)})_{k,k-2} = (B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)_{k,k-2}, \tag{65}$$

since right-multiplication by $U_j^*$ only affects columns $j$ and $j-1$. We now observe that

$$B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^* = U_{k+1} U_{k+2} \cdots U_n A U_n^* U_{n-1}^* \cdots U_{k+1}^* \tag{66}$$

is Hermitian, so from (65) we have that

$$b_{k,k-2}^{(k+1)} = \overline{(B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)}_{k-2,k}. \tag{67}$$

By the induction hypothesis,

$$\widehat{b}_{k-2,\ell}^{(k+1)} = -\widehat{p}_{k-2}^{(k+1)} q_\ell^* \tag{68}$$

and

$$|\widehat{b}_{k-2,\ell}^{(k+1)} - b_{k-2,\ell}^{(k+1)}| \lesssim \|A\| u, \tag{69}$$

for all $\ell = k, k+1, \ldots, n$. Thus,

$$\left|(-\widehat{p}_{k-2}^{(k+1)} q^* U_n^* U_{n-1}^* \cdots U_{k+1}^*)_k - (B^{(k+1)} U_n^* U_{n-1}^* \cdots U_{k+1}^*)_{k-2,k}\right| \lesssim \|A\| u. \tag{70}$$

Combining (70) with (67),

$$\left|-\widehat{p}_{k-2}^{(k+1)} \widetilde{q}_k^{(k+1)*} - \overline{b_{k,k-2}^{(k+1)}}\right| \lesssim \|A\| u, \tag{71}$$

where $\widetilde{q}^{(k+1)} = U_{k+1} U_{k+2} \cdots U_n q$. From Line 14 we have

$$\widehat{\widetilde{q}}^{(k+1)} = \mathit{fl}(\widehat{U}_{k+1} \widehat{U}_{k+2} \cdots \widehat{U}_n q), \tag{72}$$

and, by repeated application of Lemma 2.5,

$$|\widehat{\widetilde{q}}_k^{(k+1)} - \widetilde{q}_k^{(k+1)}| \lesssim \left(\sqrt{|q_k|^2 + |q_{k+1}|^2 + \cdots + |q_n|^2}\right) u. \tag{73}$$

Thus,

$$\begin{aligned} |\widehat{p}_{k-2}^{(k+1)} \widehat{\widetilde{q}}_k^{(k+1)*} - \widehat{p}_{k-2}^{(k+1)} \widetilde{q}_k^{(k+1)*}| &\lesssim \left(\sqrt{|\widehat{p}_{k-2}^{(k+1)} q_k^*|^2 + |\widehat{p}_{k-2}^{(k+1)} q_{k+1}^*|^2 + \cdots + |\widehat{p}_{k-2}^{(k+1)} q_n^*|^2}\right) u \\ &= \left(\sqrt{|\widehat{b}_{k-2,k}^{(k+1)}|^2 + |\widehat{b}_{k-2,k+1}^{(k+1)}|^2 + \cdots + |\widehat{b}_{k-2,n}^{(k+1)}|^2}\right) u \\ &\lesssim \|A\| u. \end{aligned} \tag{74}$$

where the first equality follows by (68) and the second inequality by (69). Combining (71) and (74),

$$\left|-\widehat{p}_{k-2}^{(k+1)}\widehat{\widetilde{q}}_{k}^{(k+1)*}-\overline{b_{k,k-2}^{(k+1)}}\right|\lesssim\|A\|u, \tag{75}$$

or, equivalently,

$$\left|-\widehat{\widetilde{q}}_{k}^{(k+1)}\widehat{p}_{k-2}^{(k+1)*}-b_{k,k-2}^{(k+1)}\right|\lesssim\|A\|u. \tag{76}$$

Finally, since, by the induction hypothesis,

$$\left|\widehat{\gamma}_{k-2}^{(k+1)}-b_{k-1,k-2}^{(k+1)}\right|\lesssim\|A\|u, \tag{77}$$

we use (76) and (77) and apply Lemma 2.5 to (63) to find that $|\widehat{\gamma}_{k-2}^{(k)}-b_{k-1,k-2}^{(k)}|\lesssim\|A\|u$. In Line 8, we have

$$\begin{bmatrix}\widehat{d}_{k-1}^{(k)}\\ \widehat{\gamma}_{k-1}^{(k)}\end{bmatrix}=fl\left(\widehat{Q}_k\begin{bmatrix}\widehat{d}_{k-1}^{(k+1)}\\ \widehat{\gamma}_{k-1}^{(k+1)}\end{bmatrix}\right). \tag{78}$$

By the induction hypothesis, $|\widehat{d}_{k-1}^{(k+1)}-b_{k-1,k-1}^{(k+1)}|\lesssim\|A\|u$ and $|\widehat{\gamma}_{k-1}^{(k+1)}-b_{k,k-1}^{(k+1)}|\lesssim\|A\|u$. Thus, another application of Lemma 2.5 shows that $|\widehat{d}_{k-1}^{(k)}-b_{k-1,k-1}^{(k)}|\lesssim\|A\|u$ and $|\widehat{\gamma}_{k-1}^{(k)}-b_{k,k-1}^{(k)}|\lesssim\|A\|u$. In Line 9, we have

$$\begin{bmatrix}\widehat{\beta}_{k-1}^{(k)}\\ \widehat{d}_{k}^{(k)}\end{bmatrix}=fl\left(\widehat{Q}_k\begin{bmatrix}\widehat{\beta}_{k-1}^{(k+1)}\\ \widehat{d}_{k}^{(k+1)}\end{bmatrix}\right). \tag{79}$$

By the induction hypothesis, $|\widehat{\beta}_{k-1}^{(k+1)}-b_{k-1,k}^{(k+1)}|\lesssim\|A\|u$ and $|\widehat{d}_{k}^{(k+1)}-b_{k,k}^{(k+1)}|\lesssim\|A\|u$, so it follows from Lemma 2.5 that $|\widehat{\beta}_{k-1}^{(k)}-b_{k-1,k}^{(k)}|\lesssim\|A\|u$ and $|\widehat{d}_{k}^{(k)}-b_{k,k}^{(k)}|\lesssim\|A\|u$. In Line 10, we then have

$$\begin{bmatrix}\widehat{p}_{k-1}^{(k)\dagger}\\ \widehat{p}_{k}^{(k)}\end{bmatrix}=fl\left(\widehat{Q}_k\begin{bmatrix}\widehat{p}_{k-1}^{(k+1)}\\ \widehat{p}_{k}^{(k+1)}\end{bmatrix}\right), \tag{80}$$

where $\widehat{p}_{k-1}^{(k)\dagger}$ is a temporary value that will be corrected later. By the induction hypothesis, $|\widehat{p}_{k-1}^{(k+1)}-p_{k-1}^{(k+1)}|\lesssim\|p\|u$ and $|\widehat{p}_{k}^{(k+1)}-p_{k}^{(k+1)}|\lesssim\|p\|u$, so it follows from Lemma 2.5 that $|\widehat{p}_{k-1}^{(k)\dagger}-p_{k-1}^{(k)}|\lesssim\|p\|u$ and $|\widehat{p}_{k}^{(k)}-p_{k}^{(k)}|\lesssim\|p\|u$. Define $\mathring{\beta}_{k-1}^{(k)}$ and $\mathring{d}_{k}^{(k)}$ by the formula

$$\begin{bmatrix}\mathring{\beta}_{k-1}^{(k)}\\ \mathring{d}_{k}^{(k)}\end{bmatrix}=Q_k\begin{bmatrix}\widehat{\beta}_{k-1}^{(k+1)}\\ \widehat{d}_{k}^{(k+1)}\end{bmatrix}, \tag{81}$$

and define $\mathring{p}_{k-1}^{(k)}$ and $\mathring{p}_{k}^{(k)}$ by

$$\begin{bmatrix}\mathring{p}_{k-1}^{(k)}\\ \mathring{p}_{k}^{(k)}\end{bmatrix}=Q_k\begin{bmatrix}\widehat{p}_{k-1}^{(k+1)}\\ \widehat{p}_{k}^{(k+1)}\end{bmatrix}. \tag{82}$$

Clearly,

$$\mathring{\beta}_{k-1}^{(k)} + \mathring{p}_{k-1}^{(k)} q_k^* = 0, \tag{83}$$

by the definition of $Q_k$ (see (62)). Also, by Lemma 2.5, we have that $|\mathring{\beta}_{k-1}^{(k)} - \widehat{\beta}_{k-1}^{(k)}| \lesssim \left(\sqrt{|\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2}\right) u \leq \|A\| u$ and $|\mathring{p}_{k-1}^{(k)} - \widehat{p}_{k-1}^{(k)\dagger}| \lesssim \left(\sqrt{|\widehat{p}_{k-1}^{(k+1)}|^2 + |\widehat{p}_k^{(k+1)}|^2}\right) u \leq \|p\| u$. In Line 12, we apply a correction to $\widehat{p}_{k-1}^{(k)\dagger}$, so that

$$\widehat{p}_{k-1}^{(k)} = \begin{cases} -\widehat{\beta}_{k-1}^{(k)}/q_k^* & \text{if } |\widehat{p}_{k-1}^{(k+1)} q_k^*|^2 + |\widehat{p}_k^{(k+1)} q_k^*|^2 > |\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2, \\ \widehat{p}_{k-1}^{(k)\dagger} & \text{if } |\widehat{p}_{k-1}^{(k+1)} q_k^*|^2 + |\widehat{p}_k^{(k+1)} q_k^*|^2 \leq |\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2. \end{cases} \tag{84}$$

Thus, if $|\widehat{p}_{k-1}^{(k+1)} q_k^*|^2 + |\widehat{p}_k^{(k+1)} q_k^*|^2 > |\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2$, then

$$|\widehat{p}_{k-1}^{(k)} - p_{k-1}^{(k)}| = |-\widehat{\beta}_{k-1}^{(k)}/q_k^* - p_{k-1}^{(k)}|. \tag{85}$$

We then observe that

$$|-\mathring{\beta}_{k-1}^{(k)}/q_k^* - p_{k-1}^{(k)}| = |\mathring{p}_{k-1}^{(k)} - p_{k-1}^{(k)}| \lesssim \|p\| u, \tag{86}$$

and

$$\begin{aligned} |-\mathring{\beta}_{k-1}^{(k)}/q_k^* + \widehat{\beta}_{k-1}^{(k)}/q_k^*| &= \frac{1}{|q_k^*|} |\widehat{\beta}_{k-1}^{(k)} - \mathring{\beta}_{k-1}^{(k)}| \\ &\lesssim \frac{\sqrt{|\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2}}{|q_k^*|} u \\ &\leq \left(\sqrt{|\widehat{p}_{k-1}^{(k+1)}|^2 + |\widehat{p}_k^{(k+1)}|^2}\right) u \\ &\leq \|p\| u. \end{aligned} \tag{87}$$

Finally, combining (85), (86), and (87), we find that $|\widehat{p}_{k-1}^{(k)} - p_{k-1}^{(k)}| \lesssim \|p\| u$. Furthermore,

$$|-\widehat{p}_{k-1}^{(k)} q_k^* - b_{k-1,k}^{(k)}| = |\widehat{\beta}_{k-1}^{(k)} - b_{k-1,k}^{(k)}| \lesssim \|A\| u. \tag{88}$$

If, conversely, $|\widehat{p}_{k-1}^{(k+1)} q_k^*|^2 + |\widehat{p}_k^{(k+1)} q_k^*|^2 \leq |\widehat{\beta}_{k-1}^{(k+1)}|^2 + |\widehat{d}_k^{(k+1)}|^2$, then

$$|\widehat{p}_{k-1}^{(k)} - p_{k-1}^{(k)}| = |\widehat{p}_{k-1}^{(k)\dagger} - p_{k-1}^{(k)}| \lesssim \|p\| u. \tag{89}$$

Next, we observe that

$$|-\widehat{p}_{k-1}^{(k)} q_k^* - b_{k-1,k}^{(k)}| = |-\widehat{p}_{k-1}^{(k)\dagger} q_k^* - b_{k-1,k}^{(k)}|. \tag{90}$$

Since

$$|-\mathring{p}_{k-1}^{(k)} q_k^* - b_{k-1,k}^{(k)}| = |\mathring{\beta}_{k-1}^{(k)} - b_{k-1,k}^{(k)}| \lesssim \|A\| u, \tag{91}$$

and

$$\begin{aligned}
\big|-\mathring{p}_{k-1}^{(k)}q_k^* + \widehat{p}_{k-1}^{(k)\dagger}q_k^*\big| &= |q_k^*|\big|-\mathring{p}_{k-1}^{(k)} + \widehat{p}_{k-1}^{(k)\dagger}\big| \\
&\lesssim |q_k^*|\Big(\sqrt{\big|\widehat{p}_{k-1}^{(k+1)}\big|^2 + \big|\widehat{p}_k^{(k+1)}\big|^2}\Big)u \\
&\leq \Big(\sqrt{\big|\widehat{\beta}_{k-1}^{(k+1)}\big|^2 + \big|\widehat{d}_k^{(k+1)}\big|^2}\Big)u \\
&\leq \|A\|u, \qquad (92)
\end{aligned}$$

we combine (90), (91), and (92) to see that $|-\widehat{p}_{k-1}^{(k)}q_k^* - b_{k-1,k}^{(k)}| \lesssim \|A\|u$.

Now all that's left is to show that $|-\widehat{p}_{k-1}^{(k)}q_\ell^* - b_{k-1,\ell}^{(k)}| \lesssim \|A\|u$ and $|-\widehat{p}_k^{(k)}q_\ell^* - b_{k,\ell}^{(k)}| \lesssim \|A\|u$, for all $\ell = k+1, k+2, \ldots, n$. By the induction hypothesis,

$$\big|-\widehat{p}_{k-1}^{(k+1)}q_\ell^* - b_{k-1,\ell}^{(k+1)}\big| \lesssim \|A\|u \qquad (93)$$

and

$$\big|-\widehat{p}_k^{(k+1)}q_\ell^* - b_{k,\ell}^{(k+1)}\big| \lesssim \|A\|u, \qquad (94)$$

for all $\ell = k+1, k+2, \ldots, n$. Multiplying (82) by $q_\ell^*$, we have

$$\begin{bmatrix} \mathring{p}_{k-1}^{(k)}q_\ell^* \\ \mathring{p}_k^{(k)}q_\ell^* \end{bmatrix} = Q_k \begin{bmatrix} \widehat{p}_{k-1}^{(k+1)}q_\ell^* \\ \widehat{p}_k^{(k+1)}q_\ell^* \end{bmatrix}, \qquad (95)$$

which, combined with (93) and (94), means that

$$\big|-\mathring{p}_{k-1}^{(k)}q_\ell^* - b_{k-1,\ell}^{(k)}\big| \lesssim \|A\|u \qquad (96)$$

and

$$\big|-\mathring{p}_k^{(k)}q_\ell^* - b_{k,\ell}^{(k)}\big| \lesssim \|A\|u, \qquad (97)$$

for all $\ell = k+1, k+2, \ldots, n$. It is not difficult to show (see (87)) that

$$|\mathring{p}_{k-1}^{(k)} - \widehat{p}_{k-1}^{(k)}| \lesssim \Big(\sqrt{\big|\widehat{p}_{k-1}^{(k+1)}\big|^2 + \big|\widehat{p}_k^{(k+1)}\big|^2}\Big)u \qquad (98)$$

and

$$|\mathring{p}_k^{(k)} - \widehat{p}_k^{(k)}| \lesssim \Big(\sqrt{\big|\widehat{p}_{k-1}^{(k+1)}\big|^2 + \big|\widehat{p}_k^{(k+1)}\big|^2}\Big)u, \qquad (99)$$

from which it follows that

$$|\mathring{p}_{k-1}^{(k)}q_\ell^* - \widehat{p}_{k-1}^{(k)}q_\ell^*| \lesssim \Big(\sqrt{\big|\widehat{p}_{k-1}^{(k+1)}q_\ell^*\big|^2 + \big|\widehat{p}_k^{(k+1)}q_\ell^*\big|^2}\Big)u \lesssim \|A\|u \qquad (100)$$

and

$$|\mathring{p}_k^{(k)}q_\ell^* - \widehat{p}_k^{(k)}q_\ell^*| \lesssim \Big(\sqrt{\big|\widehat{p}_{k-1}^{(k+1)}q_\ell^*\big|^2 + \big|\widehat{p}_k^{(k+1)}q_\ell^*\big|^2}\Big)u \lesssim \|A\|u, \qquad (101)$$

for all $\ell = k+1, k+2, \ldots, n$, where the second inequality follows from (93) and (94). Finally, combining (96), (97), (100), and (101), we find that $|-\widehat{p}_{k-1}^{(k)} q_\ell^* - b_{k-1,\ell}^{(k)}| \lesssim \|A\| u$ and $|-\widehat{p}_k^{(k)} q_\ell^* - b_{k,\ell}^{(k)}| \lesssim \|A\| u$, for all $\ell = k+1, k+2, \ldots, n$, and we are done.

■

The following lemma bounds the forward error of Algorithm 2 (the rotation back to Hessenberg form).

**Lemma 4.2.** *Suppose that $B \in \mathbb{C}^{n\times n}$ and $p, q \in \mathbb{C}^n$. Suppose further that $B + pq^*$ is lower triangular, and let $d$ and $\gamma$ denote the diagonal and subdiagonal of $B$, respectively. Suppose that $Q_2, Q_3, \ldots, Q_n \in \mathrm{SU}(2)$, and suppose that Algorithm 2 is carried out in floating point arithmetic, using $d$, $\gamma$, $p$, $q$, and $Q_2, Q_3, \ldots, Q_n$ as inputs. Suppose finally that $\widehat{\underline{d}}$, $\widehat{\underline{\beta}}$, and $\widehat{\underline{q}}$ are the outputs generated by Algorithm 2, and define the upper triangular part of the matrix $\widehat{\underline{A}} \in \mathbb{C}^{n\times n}$ by the formula*

$$\widehat{\underline{a}}_{i,j} = \begin{cases} -p_i \widehat{\underline{q}}_j^* & \text{if } j > i+1, \\ \widehat{\underline{\beta}}_i & \text{if } j = i+1, \\ \widehat{\underline{d}}_i & \text{if } j = i, \end{cases} \tag{102}$$

*where $\widehat{\underline{a}}_{i,j}$ denotes the $(i,j)$-th entry of $\widehat{\underline{A}}$. Let $U_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $Q_k$. Define $U \in \mathbb{C}^{n\times n}$ by the formula $U = U_2 U_3 \cdots U_n$, and let $\underline{A} = BU^*$ and $\underline{q} = Uq$. Then*

$$\|\widehat{\underline{A}} - \underline{A}\|_T \lesssim \|B\|_H u \tag{103}$$

*and*

$$\|\widehat{\underline{q}} - \underline{q}\| \lesssim \|q\| u, \tag{104}$$

*where $\|\cdot\|_T$ denotes the square root of the sum of squares of the entries in the upper triangular part of its argument and $\|\cdot\|_H$ denotes the square root of the sum of squares of the upper Hessenberg part (see Definition 2.1).*

**Proof.** Suppose that $\widehat{d}^{(k)}$, $\widehat{\beta}^{(k)}$, and $\widehat{q}^{(k)}$ denote the computed vectors in Algorithm 2 after rotations in the positions $(n-1, n), (n-2, n-1), \ldots, (k-1, k)$. Suppose further that the upper triangular part of the matrix $\widehat{A}^{(k)} \in \mathbb{C}^{n\times n}$ is defined by the formula

$$\widehat{a}_{i,j}^{(k)} = \begin{cases} -p_i \widehat{q}_j^{(k)*} & \text{if } j > i+1 \text{ or if } j = i+1 \text{ and } j < k, \\ \widehat{\beta}_i^{(k)} & \text{if } j = i+1 \text{ and } j \geq k, \\ \widehat{d}_i^{(k)} & \text{if } j = i, \end{cases} \tag{105}$$

where $\widehat{a}_{i,j}^{(k)}$ denotes the $(i,j)$-th entry of $\widehat{A}^{(k)}$. Clearly, $\widehat{\underline{d}} = \widehat{d}^{(2)}$, $\widehat{\underline{\beta}} = \widehat{\beta}^{(2)}$, $\widehat{\underline{q}} = \widehat{q}^{(2)}$, and $\widehat{\underline{A}} = \widehat{A}^{(2)}$. Let $A^{(k)} = BU_n^* U_{n-1}^* \cdots U_k^*$ and $q^{(k)} = U_k U_{k+1} \cdots U_n q$. We will prove that $\|\widehat{A}^{(k)} - A^{(k)}\|_T \lesssim \|B\|_H u$ and $\|\widehat{q}^{(k)} - q^{(k)}\| \lesssim \|q\| u$, for each $k = n, n-1, \ldots, 2$.

Define $\widehat{d}^{(n+1)} = d$, $\widehat{q}^{(n+1)} = q$, $q^{(n+1)} = q$, and $A^{(n+1)} = B$. Obviously, $\widehat{A}^{(n+1)} = A^{(n+1)}$ and $\widehat{q}^{(n+1)} = q^{(n+1)}$, so the above statement is true for $k = n+1$. We will prove it for the cases $k = n, n-1, \ldots, 2$ by induction. In Line 2, we have

$$\begin{bmatrix} \widehat{d}_{k-1}^{(k)} \\ \widehat{\beta}_{k-1}^{(k)} \end{bmatrix} = fl\left( \widehat{\overline{Q_k}} \begin{bmatrix} \widehat{d}_{k-1}^{(k+1)} \\ -p_{k-1} \widehat{q}_k^{(k+1)*} \end{bmatrix} \right). \tag{106}$$

By the induction hypothesis, $|\widehat{d}_{k-1}^{(k+1)} - a_{k-1,k-1}^{(k+1)}| \lesssim \|B\|_H u$ and $|-p_{k-1}\widehat{q}_k^{(k+1)*} - a_{k-1,k}^{(k+1)}| \lesssim \|B\|_H u$. Since, by Lemma 2.3, $\|A^{(k+1)}\|_T \leq \|B\|_H$, an application of Lemma 2.5 gives us $|\widehat{d}_{k-1}^{(k)} - a_{k-1,k-1}^{(k)}| \lesssim \|B\|_H u$ and $|-p_{k-1}\widehat{q}_k^{(k)*} - a_{k-1,k}^{(k)}| \lesssim \|B\|_H u$. In Line 3, we have

$$\widehat{d}_k^{(k)} = \text{fl}\left(\widehat{\overline{Q}_k}\begin{bmatrix} \gamma_{k-1} \\ \widehat{d}_k^{(k+1)} \end{bmatrix}\right)_2. \tag{107}$$

We first observe that

$$\gamma_{k-1} = b_{k,k-1} = (BU_n^* U_{n-1}^* \cdots U_{k+1}^*)_{k,k-1} = a_{k,k-1}^{(k+1)}, \tag{108}$$

since right multiplication by $U_j^*$ only affects columns $j$ and $j-1$. Since, by the induction hypothesis, $|\widehat{d}_k^{(k+1)} - a_{k,k}^{(k+1)}| \lesssim \|B\|_H u$, an application of Lemma 2.5 together with the inequality $\|A^{(k+1)}\|_T \leq \|B\|_H$ gives us $|\widehat{d}_k^{(k)} - a_{k,k}^{(k)}| \lesssim \|B\|_H u$. In Line 4,

$$\begin{bmatrix} \widehat{q}_{k-1}^{(k)} \\ \widehat{q}_k^{(k)} \end{bmatrix} = \text{fl}\left(\widehat{Q}_k \begin{bmatrix} \widehat{q}_{k-1}^{(k+1)} \\ \widehat{q}_k^{(k+1)} \end{bmatrix}\right). \tag{109}$$

By the induction hypothesis, $|\widehat{q}_{k-1}^{(k+1)} - q_{k-1}^{(k+1)}| \lesssim \|q\| u$ and $|\widehat{q}_k^{(k+1)} - q_k^{(k+1)}| \lesssim \|q\| u$, so it follows from Lemma 2.5 that $|\widehat{q}_{k-1}^{(k)} - q_{k-1}^{(k)}| \lesssim \|q\| u$ and $|\widehat{q}_k^{(k)} - q_k^{(k)}| \lesssim \|q\| u$.

All that's left now is to prove that $|-p_\ell \widehat{q}_{k-1}^{(k)*} - a_{\ell,k-1}^{(k)}| \lesssim \|B\|_H u$ and $|-p_\ell \widehat{q}_k^{(k)*} - a_{\ell,k}^{(k)}| \lesssim \|B\|_H u$, for all for all $\ell = 1, 2, \ldots, k-2$. By the induction hypothesis,

$$|-p_\ell \widehat{q}_{k-1}^{(k+1)*} - a_{\ell,k-1}^{(k+1)}| \lesssim \|B\|_H u \tag{110}$$

and

$$|-p_\ell \widehat{q}_k^{(k+1)*} - a_{\ell,k}^{(k+1)}| \lesssim \|B\|_H u, \tag{111}$$

for all $\ell = 1, 2, \ldots, k-2$. Define $\mathring{q}_{k-1}^{(k)}$ and $\mathring{q}_k^{(k)}$ by

$$\begin{bmatrix} \mathring{q}_{k-1}^{(k)} \\ \mathring{q}_k^{(k)} \end{bmatrix} = Q_k \begin{bmatrix} \widehat{q}_{k-1}^{(k+1)} \\ \widehat{q}_k^{(k+1)} \end{bmatrix}. \tag{112}$$

Multiplying (112) by $p_\ell^*$, we have

$$\begin{bmatrix} \mathring{q}_{k-1}^{(k)} p_\ell^* \\ \mathring{q}_k^{(k)} p_\ell^* \end{bmatrix} = Q_k \begin{bmatrix} \widehat{q}_{k-1}^{(k+1)} p_\ell^* \\ \widehat{q}_k^{(k+1)} p_\ell^* \end{bmatrix}, \tag{113}$$

which, combined with (110) and (111) and the fact that $\|A^{(k+1)}\|_T \leq \|B\|_H$, means that

$$\left|-p_\ell \mathring{q}_{k-1}^{(k)*} - a_{\ell,k-1}^{(k)}\right| \lesssim \|B\|_H u \tag{114}$$

and

$$\left|-p_\ell \mathring{q}_k^{(k)*} - a_{\ell,k}^{(k)}\right| \lesssim \|B\|_H u, \tag{115}$$

for all $\ell = 1, 2, \ldots, k-2$. By Lemma 2.5,

$$|\mathring{q}_{k-1}^{(k)} - \widehat{q}_{k-1}^{(k)}| \lesssim \left(\sqrt{|\widehat{q}_{k-1}^{(k+1)}|^2 + |\widehat{q}_{k}^{(k+1)}|^2}\right) u \tag{116}$$

and

$$|\mathring{q}_{k}^{(k)} - \widehat{q}_{k}^{(k)}| \lesssim \left(\sqrt{|\widehat{q}_{k-1}^{(k+1)}|^2 + |\widehat{q}_{k}^{(k+1)}|^2}\right) u, \tag{117}$$

from which it follows that

$$|\mathring{q}_{k-1}^{(k)} p_\ell^* - \widehat{q}_{k-1}^{(k)} p_\ell^*| \lesssim \left(\sqrt{|\widehat{q}_{k-1}^{(k+1)} p_\ell^*|^2 + |\widehat{q}_{k}^{(k+1)} p_\ell^*|^2}\right) u \lesssim \|B\|_H u \tag{118}$$

and

$$|\mathring{q}_{k}^{(k)} p_\ell^* - \widehat{q}_{k}^{(k)} p_\ell^*| \lesssim \left(\sqrt{|\widehat{q}_{k-1}^{(k+1)} p_\ell^*|^2 + |\widehat{q}_{k}^{(k+1)} p_\ell^*|^2}\right) u \lesssim \|B\|_H u, \tag{119}$$

for all $\ell = 1, 2, \ldots, k-2$, where the second inequality follows from (110) and (111) and the inequality $\|A^{(k+1)}\|_T \leq \|B\|_H$. Finally, combining (114), (115), (118), and (119), we find that $|-p_\ell \widehat{q}_{k-1}^{(k)*} - a_{\ell,k-1}^{(k)}| \lesssim \|B\|_H u$ and $|-p_\ell \widehat{q}_{k}^{(k)*} - a_{\ell,k}^{(k)}| \lesssim \|B\|_H u$, for all for all $\ell = 1, 2, \ldots, k-2$, and we are done.

■

The following theorem bounds the forward errors of a full sweep of our $QR$ algorithm, and is the principal result of this subsection.

**Theorem 4.3.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is a Hermitian matrix, that $p, q \in \mathbb{C}^n$, and that $A + pq^*$ is lower Hessenberg. Let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that Algorithm 1 is carried out in floating point arithmetic, and let $Q_2, Q_3, \ldots, Q_n \in \mathrm{SU}(2)$ be the unitary matrices generated by an exact step of Line 4 of Algorithm 1 applied to the computed vectors at that step. Let $U_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $Q_k$, and define $U \in \mathbb{C}^{n\times n}$ by the formula $U = U_2 U_3 \cdots U_n$. Suppose that Algorithm 2 is then carried out in floating point arithmetic, using the outputs of Algorithm 1 as inputs. Suppose finally that $\widehat{\underline{p}}$ is an output of Algorithm 1 and $\widehat{\underline{q}}$, $\widehat{\underline{d}}$, and $\widehat{\underline{\beta}}$ are all outputs of Algorithm 2, and define the matrix $\widehat{\underline{A}}$ by the formula*

$$\widehat{\underline{a}}_{i,j} = \begin{cases} -\widehat{\underline{p}}_i \widehat{\underline{q}}_j^* & \text{if } j > i+1 \\ \widehat{\underline{\beta}}_i & \text{if } j = i+1 \\ \widehat{\underline{d}}_i & \text{if } j = i \\ \overline{(\widehat{\underline{\beta}}_j)} & \text{if } j = i-1 \\ -\widehat{\underline{q}}_j \widehat{\underline{p}}_i^* & \text{if } j < i-1 \end{cases} \tag{120}$$

*where $\widehat{\underline{a}}_{i,j}$ denotes the $(i,j)$-th entry of $\widehat{\underline{A}}$. Let $\underline{A} = UAU^*$, $\underline{p} = Up$, and $\underline{q} = Uq$. Then*

$$\|\widehat{\underline{A}} - \underline{A}\| \lesssim \|A\| u, \tag{121}$$

$$\|\widehat{\underline{p}} - \underline{p}\| \lesssim \|p\| u, \tag{122}$$

*and*

$$\|\widehat{\underline{q}} - \underline{q}\| \lesssim \|q\| u. \tag{123}$$

**Proof.** Suppose that $\widehat{B}$ (defined by (47)), $\widehat{\underline{p}}$, and $\widehat{Q}_2, \widehat{Q}_3, \ldots, \widehat{Q}_n \in \mathbb{C}^{n\times n}$ are outputs of Algorithm 1. Let $B = UA$ and $\underline{p} = Up$. By Lemma 4.1,

$$\|\widehat{B} - B\|_H \lesssim \|A\| u \tag{124}$$

and

$$\|\widehat{\underline{p}} - \underline{p}\| \lesssim \|p\| u, \tag{125}$$

where $\|\cdot\|_H$ denotes the square root of the sum of squares of the entries in the upper Hessenberg part of its argument (see Definition 2.1). Now suppose that $\widehat{B}$, $\widehat{\underline{p}}$, $q$, and $\widehat{Q}_2, \widehat{Q}_3, \ldots, \widehat{Q}_n \in \mathbb{C}^{n\times n}$ are used as inputs to Algorithm 2. Let $\widehat{U}_k \in \mathbb{C}^{n\times n}$, $k = 2, 3, \ldots, n$, denote the matrices that rotate the $(k-1, k)$-plane by $\widehat{Q}_k$, and define $\widehat{U} \in \mathbb{C}^{n\times n}$ by the formula $\widehat{U} = \widehat{U}_2\widehat{U}_3 \cdots \widehat{U}_n$. Let $\underset{\sim}{\underline{A}} = \widehat{B}\widehat{U}^*$ and $\underline{q} = \widehat{U}q$, and observe that the upper triangular part of $\underset{\sim}{\underline{A}}$ is well-defined due to Lemma 2.3. By Lemma 4.2 we have that

$$\|\widehat{\underline{A}} - \underset{\sim}{\underline{A}}\|_T \lesssim \|\widehat{B}\|_H u \tag{126}$$

and

$$\|\widehat{\underline{q}} - \underline{q}\| \lesssim \|q\| u, \tag{127}$$

where $\|\cdot\|_T$ denotes the square root of the sum of squares of the entries in the upper triangular part of its argument and $\|\cdot\|_H$ denotes the square root of the sum of squares of the upper Hessenberg part (see Definition 2.1). Let $\underline{A} = BU^* = UAU^*$ and let $\underline{q} = Uq$. We observe that

$$\begin{aligned}
&\|\underset{\sim}{\underline{A}} - \underline{A}\|_T = \|\widehat{B}\widehat{U}^* - BU^*\|_T \\
&\leq \|\widehat{B}\widehat{U}^* - B\widehat{U}^*\|_T + \|B\widehat{U}^* - BU^*\|_T \\
&= \|(\widehat{B} - B)\widehat{U}^*\|_T + \|B(\widehat{U}^* - U^*)\|_T \\
&\lesssim \|A\| u,
\end{aligned} \tag{128}$$

where the last inequality follows from (124) and the fact that $\|\widehat{U} - U\| \lesssim u$. Since, clearly, $\|\widehat{B}\|_H u \lesssim \|A\| u$, we combine (126) and (128) to get

$$\|\widehat{\underline{A}} - \underline{A}\|_T \lesssim \|A\| u. \tag{129}$$

Now we observe that, since both $\underline{A} = UAU^*$ and $\widehat{\underline{A}}$ are Hermitian,

$$\|\widehat{\underline{A}} - \underline{A}\| \lesssim \|A\| u. \tag{130}$$

Next, we observe that

$$\begin{aligned}\|\underset{\sim}{q} - \underline{q}\| &= \|\widehat{U}q - Uq\| \\ &= \|(\widehat{U} - U)q\| \\ &\lesssim \|q\|u,\end{aligned} \tag{131}$$

so, combining (127) and (131), we have

$$\|\widehat{\underline{q}} - \underline{q}\| \lesssim \|q\|u, \tag{132}$$

and we are done.

■

## 4.2 Backward Error Analysis of the $QR$ Algorithms

Suppose that $A$ is Hermitian and $A + pq^*$ is lower Hessenberg. In this section, we prove in Theorems 4.6 and 4.7 that the backward errors in $A$, $p$, and $q$ of both our explicit unshifted $QR$ algorithm (see Algorithm 3) and explicit shifted $QR$ algorithm (see Algorithm 4) are proportional to $\|A\|u$, $\|p\|u$, and $\|q\|u$, respectively.

The following lemma states that a single iteration of our QR algorithm is componentwise backward stable.

**Lemma 4.4.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is a Hermitian matrix, that $p, q \in \mathbb{C}^n$, and that $A + pq^*$ is lower Hessenberg. Let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that a single iteration of our QR algorithm (Algorithm 1 followed by Algorithm 2) is carried out in floating point arithmetic, and let $\widehat{\underline{p}}$, $\widehat{\underline{q}}$, $\widehat{\underline{d}}$, and $\widehat{\underline{\beta}}$ denote the outputs of the algorithm. Define the matrix $\widehat{\underline{A}}$ by the formula (120). Then there exists a unitary matrix $U \in \mathbb{C}^{n\times n}$, a matrix $\delta A \in \mathbb{C}^{n\times n}$, and vectors $\delta p, \delta q \in \mathbb{C}^n$, such that*

$$\widehat{\underline{A}} = U(A + \delta A)U^*, \tag{133}$$

$$\widehat{\underline{p}} = U(p + \delta p), \tag{134}$$

*and*

$$\widehat{\underline{q}} = U(q + \delta q), \tag{135}$$

*where $\|\delta A\| \lesssim \|A\|u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$.*

**Proof.** Let $U \in \mathbb{C}^{n\times n}$ be the unitary matrix defined in the statement of Theorem 4.3, and let $\underline{A} = UAU^*$, $\underline{p} = Up$, and $\underline{q} = Uq$. By Theorem 4.3,

$$\widehat{\underline{A}} = \underline{A} + \underline{\delta A}, \tag{136}$$

where $\|\underline{\delta A}\| \lesssim \|A\|u$. Thus,

$$\widehat{\underline{A}} = UAU^* + \underline{\delta A} = U(A + \delta A)U^*, \tag{137}$$

where $\delta A = U^* \underline{\delta A} U$. Since $U$ is unitary, clearly $\|\delta A\| \lesssim \|A\| u$. Likewise, by Theorem 4.3,

$$\widehat{\underline{p}} = \underline{p} + \underline{\delta p}, \tag{138}$$

where $\|\underline{\delta p}\| \lesssim \|p\| u$, so

$$\widehat{\underline{p}} = U(p + \delta p), \tag{139}$$

where $\delta p = U^* \underline{\delta p}$ and $\|\delta p\| \lesssim \|p\| u$. Similarly,

$$\widehat{\underline{q}} = U(q + \delta q), \tag{140}$$

where $\|\delta q\| \lesssim \|q\| u$.

■

The following lemma states that repeated iterations of our QR algorithm are componentwise backward stable.

**Lemma 4.5.** *Suppose that $A \in \mathbb{C}^{n \times n}$ is a Hermitian matrix, that $p, q \in \mathbb{C}^n$, and that $A + pq^*$ is lower Hessenberg. Let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that $k$ iterations of our QR algorithm (Algorithm 1 followed by Algorithm 2) are carried out in floating point arithmetic, and let $\widehat{p}^{(k)}$, $\widehat{q}^{(k)}$, $\widehat{d}^{(k)}$, and $\widehat{\beta}^{(k)}$ denote the outputs of the algorithm. Define the matrix $\widehat{A}^{(k)}$ by the formula (120), making the obvious substitutions. Then there exists a unitary matrix $U \in \mathbb{C}^{n \times n}$, a matrix $\delta A \in \mathbb{C}^{n \times n}$, and vectors $\delta p, \delta q \in \mathbb{C}^n$, such that*

$$\widehat{A}^{(k)} = U(A + \delta A)U^*, \tag{141}$$

$$\widehat{p}^{(k)} = U(p + \delta p), \tag{142}$$

*and*

$$\widehat{q}^{(k)} = U(q + \delta q), \tag{143}$$

*where $\|\delta A\| \lesssim \|A\| u$, $\|\delta p\| \lesssim \|p\| u$, and $\|\delta q\| \lesssim \|q\| u$.*

**Proof.** We will prove this statement only for the matrix $\widehat{A}^{(k)}$, since the proofs for $\widehat{p}^{(k)}$ and $\widehat{q}^{(k)}$ are essentially identical. By repeated application of Lemma 4.4, we know that there exist unitary matrices $U^{(1)}, U^{(2)}, \ldots, U^{(k)}$ and matrices $\delta A^{(0)}, \delta\widehat{A}^{(1)}, \delta\widehat{A}^{(2)}, \ldots, \delta\widehat{A}^{(k-1)}$ such that

$$\widehat{A}^{(k)} = U^{(k)}(\widehat{A}^{(k-1)} + \delta\widehat{A}^{(k-1)})U^{(k)*}, \tag{144}$$

$$\widehat{A}^{(k-1)} = U^{(k-1)}(\widehat{A}^{(k-2)} + \delta\widehat{A}^{(k-2)})U^{(k-1)*}, \tag{145}$$

$$\vdots$$

$$\widehat{A}^{(2)} = U^{(2)}(\widehat{A}^{(1)} + \delta\widehat{A}^{(1)})U^{(2)*}, \tag{146}$$

$$\widehat{A}^{(1)} = U^{(1)}(A + \delta A^{(0)})U^{(1)*}, \tag{147}$$

where $\|\delta A^{(0)}\| \lesssim \|A\|u$ and $\|\delta\widehat{A}^{(\ell)}\| \lesssim \|A\|u$, for $\ell = 1, 2, \ldots, k-1$. Combining (144)–(147) and expanding, we find that

$$\begin{aligned}\widehat{A}^{(k)} &= U^{(k)}U^{(k-1)}\cdots U^{(1)}AU^{(1)*}U^{(2)*}\cdots U^{(k)*} + U^{(k)}\delta\widehat{A}^{(k-1)}U^{(k)*} \\ &+ U^{(k)}U^{(k-1)}\delta\widehat{A}^{(k-2)}U^{(k-1)*}U^{(k)*} + \cdots \\ &+ U^{(k)}U^{(k-1)}\cdots U^{(1)}\delta A^{(0)}U^{(1)*}\cdots U^{(k-1)*}U^{(k)*}. \end{aligned} \tag{148}$$

Letting $U = U^{(k)}U^{(k-1)}\cdots U^{(1)}$, this becomes

$$\begin{aligned}\widehat{A}^{(k)} &= UAU^* + U^{(k)}\delta\widehat{A}^{(k-1)}U^{(k)*} \\ &+ U^{(k)}U^{(k-1)}\delta\widehat{A}^{(k-2)}U^{(k-1)*}U^{(k)*} + \cdots \\ &+ U^{(k)}U^{(k-1)}\cdots U^{(1)}\delta A^{(0)}U^{(1)*}\cdots U^{(k-1)*}U^{(k)*}. \end{aligned} \tag{149}$$

Suppose now that the matrix $\delta A$ is defined by

$$\begin{aligned}\delta A = U^*\big(&U^{(k)}\delta\widehat{A}^{(k-1)}U^{(k)*} + U^{(k)}U^{(k-1)}\delta\widehat{A}^{(k-2)}U^{(k-1)*}U^{(k)*} + \cdots \\ &+ U^{(k)}U^{(k-1)}\cdots U^{(1)}\delta A^{(0)}U^{(1)*}\cdots U^{(k-1)*}U^{(k)*}\big)U. \end{aligned} \tag{150}$$

Clearly, $\|\delta A\| \lesssim \|A\|u$. Combining (149) and (150), we have

$$\widehat{A}^{(k)} = U(A + \delta A)U^*, \tag{151}$$

and we are done.

■

The following theorem states that our explicit unshifted QR algorithm is componentwise backward stable.

**Theorem 4.6** (Explicit unshifted QR)**.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is a Hermitian matrix, that $p, q \in \mathbb{C}^n$, and that $A + pq^*$ is lower Hessenberg. Let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that Algorithm 3 is carried out in floating point arithmetic with $\epsilon \lesssim \|A\|u$, and let $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ denote the outputs. Then there exist a matrix $\delta A \in \mathbb{C}^{n\times n}$ and vectors $\delta p, \delta q \in \mathbb{C}^n$ such that $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ are the exact eigenvalues of the matrix*

$$(A + \delta A) + (p + \delta p)(q + \delta q)^*, \tag{152}$$

*where $\|\delta A\| \lesssim \|A\|u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$.*

**Proof.** Suppose that we carry out QR iterations until the entry in the $(1, 2)$ position is less than $\epsilon$ in absolute value. Let $\widehat{d}^{(1)}$, $\widehat{\beta}^{(1)\dagger}$, $\widehat{p}^{(1)}$, and $\widehat{q}^{(1)}$ denote the resulting outputs, and let $\widehat{A}^{(1)\dagger}$ be the resulting matrix, defined by formula (120) (making the obvious substitutions). By Lemma 4.5, there exist a unitary matrix $U^{(1)} \in \mathbb{C}^{n\times n}$, a matrix $\delta A^{(0)\dagger} \in \mathbb{C}^{n\times n}$, and vectors $\delta p^{(0)}, \delta q^{(0)} \in \mathbb{C}^n$ such that

$$\widehat{A}^{(1)\dagger} = U^{(1)}(A + \delta A^{(0)\dagger})U^{(1)*}, \tag{153}$$

$$\widehat{p}^{(1)} = U^{(1)}(p + \delta p^{(0)}), \tag{154}$$

and

$$\widehat{q}^{(1)} = U^{(1)}(q + \delta q^{(0)}), \tag{155}$$

where $\|\delta A^{(0)\dagger}\| \lesssim \|A\|u$, $\|\delta p^{(0)}\| \lesssim \|p\|u$, and $\|\delta q^{(0)}\| \lesssim \|q\|u$. Let $\widehat{A}^{(1)}$ be equal to $\widehat{A}^{(1)\dagger}$, except that the entry in the $(1,2)$ position of $\widehat{A}^{(1)}$ is equal to $-\widehat{p}_1^{(1)}\widehat{q}_2^{(1)*}$, so that $\widehat{A}_{1,2}^{(1)} + \widehat{p}_1^{(1)}\widehat{q}_2^{(1)*} = 0$. Since $|\widehat{A}_{1,2}^{(1)\dagger} + \widehat{p}_1^{(1)}\widehat{q}_2^{(1)*}| < \epsilon$, we have

$$\|\widehat{A}^{(1)\dagger} - \widehat{A}^{(1)}\|_F < \epsilon, \tag{156}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and since $\epsilon \lesssim \|A\|u$,

$$\|\widehat{A}^{(1)\dagger} - \widehat{A}^{(1)}\| \lesssim \|A\|u. \tag{157}$$

Letting

$$\delta A^{(0)} = \delta A^{(0)\dagger} + U^{(1)*}(\widehat{A}^{(1)} - \widehat{A}^{(1)\dagger})U^{(1)} \tag{158}$$

and combining (153) and (158), we have

$$\widehat{A}^{(1)} = U^{(1)}(A + \delta A^{(0)})U^{(1)*}, \tag{159}$$

where $\|\delta A^{(0)}\| \lesssim \|A\|u$ by (157). Clearly, since $\widehat{A}_{1,2}^{(1)} + \widehat{p}_1^{(1)}\widehat{q}_2^{(1)*} = 0$ and $\widehat{A}^{(1)} + \widehat{p}^{(1)}\widehat{q}^{(1)*}$ is lower Hessenberg, $\widehat{\lambda}_1 = \widehat{A}_{1,1}^{(1)} + \widehat{p}_1^{(1)}\widehat{q}_1^{(1)*}$ is an eigenvalue of $\widehat{A}^{(1)} + \widehat{p}^{(1)}\widehat{q}^{(1)*}$. Thus, from (154), (155), and (159), we see that $\widehat{\lambda}_1$ is an eigenvalue of $(A+\delta A^{(0)})+(p+\delta p^{(0)})(q+\delta q^{(0)})^*$.

Now suppose that we deflate the matrix, and perform QR iterations on the submatrix $\widehat{A}_{2:n,2:n}^{(1)} + \widehat{p}_{2:n}^{(1)}\widehat{q}_{2:n}^{(1)*}$, until the entry in the $(1,2)$ position of the deflated matrix is less than $\epsilon$. Let $\underset{\sim}{\widehat{d}}^{(2)} \in \mathbb{C}^{n-1}$, $\underset{\sim}{\widehat{\beta}}^{(2)\dagger} \in \mathbb{C}^{n-2}$, $\underset{\sim}{\widehat{p}}^{(2)} \in \mathbb{C}^{n-1}$, and $\underset{\sim}{\widehat{q}}^{(2)} \in \mathbb{C}^{n-1}$ denote the resulting outputs, and let $\underset{\sim}{\widehat{A}}^{(2)\dagger} \in \mathbb{C}^{(n-1)\times(n-1)}$ be the resulting matrix, defined by formula (120) (again making the obvious substitutions). By Lemma 4.5, there exist a unitary matrix $\underset{\sim}{U}^{(2)} \in \mathbb{C}^{(n-1)\times(n-1)}$, a matrix $\delta\underset{\sim}{\widehat{A}}^{(1)\dagger} \in \mathbb{C}^{(n-1)\times(n-1)}$, and vectors $\delta\underset{\sim}{\widehat{p}}^{(1)}, \delta\underset{\sim}{\widehat{q}}^{(1)} \in \mathbb{C}^{n-1}$ such that

$$\underset{\sim}{\widehat{A}}^{(2)\dagger} = \underset{\sim}{U}^{(2)}(\widehat{A}_{2:n,2:n}^{(1)} + \delta\underset{\sim}{\widehat{A}}^{(1)\dagger})\underset{\sim}{U}^{(2)*}, \tag{160}$$

$$\underset{\sim}{\widehat{p}}^{(2)} = U^{(2)}(\widehat{p}_{2:n}^{(1)} + \delta\underset{\sim}{\widehat{p}}^{(1)}), \tag{161}$$

and

$$\underset{\sim}{\widehat{q}}^{(2)} = U^{(2)}(\widehat{q}_{2:n}^{(1)} + \delta\underset{\sim}{\widehat{q}}^{(1)}), \tag{162}$$

where $\|\delta\underset{\sim}{\widehat{A}}^{(1)\dagger}\| \lesssim \|A\|u$, $\|\delta\underset{\sim}{\widehat{p}}^{(1)}\| \lesssim \|p\|u$, and $\|\delta\underset{\sim}{\widehat{q}}^{(1)}\| \lesssim \|q\|u$. Like before, let $\underset{\sim}{\widehat{A}}^{(2)} \in \mathbb{C}^{(n-1)\times(n-1)}$ be equal to $\underset{\sim}{\widehat{A}}^{(2)\dagger}$, except that the entry in the $(1,2)$ position of $\underset{\sim}{\widehat{A}}^{(2)}$ is equal to $-\underset{\sim}{\widehat{p}}_1^{(2)}\underset{\sim}{\widehat{q}}_2^{(2)*}$, so that $\underset{\sim}{\widehat{A}}_{1,2}^{(2)} + \underset{\sim}{\widehat{p}}_1^{(2)}\underset{\sim}{\widehat{q}}_2^{(2)*} = 0$. Since $|\underset{\sim}{\widehat{A}}_{1,2}^{(2)\dagger} + \underset{\sim}{\widehat{p}}_1^{(2)}\underset{\sim}{\widehat{q}}_2^{(2)*}| < \epsilon$, we have

$$\|\underset{\sim}{\widehat{A}}^{(2)\dagger} - \underset{\sim}{\widehat{A}}^{(2)}\|_F < \epsilon, \tag{163}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and since $\epsilon \lesssim \|A\|u$,

$$\|\widehat{\underset{\sim}{A}}^{(2)\dagger} - \widehat{\underset{\sim}{A}}^{(2)}\| \lesssim \|A\|u. \tag{164}$$

Letting

$$\delta\widehat{\underset{\sim}{A}}^{(1)} = \delta\widehat{\underset{\sim}{A}}^{(1)\dagger} + \underset{\sim}{U}^{(2)*}(\widehat{\underset{\sim}{A}}^{(2)} - \widehat{\underset{\sim}{A}}^{(2)\dagger})\underset{\sim}{U}^{(2)}, \tag{165}$$

we have

$$\widehat{\underset{\sim}{A}}^{(2)} = \underset{\sim}{U}^{(2)}(\widehat{A}^{(1)}_{2:n,2:n} + \delta\widehat{\underset{\sim}{A}}^{(1)})\underset{\sim}{U}^{(2)*}, \tag{166}$$

where $\|\delta\widehat{\underset{\sim}{A}}^{(1)}\| \lesssim \|A\|u$. Since $\widehat{\underset{\sim}{A}}^{(2)}_{1,2} + \widehat{\underset{\sim}{p}}^{(1)}_1\widehat{\underset{\sim}{q}}^{(1)*}_2 = 0$ and $\widehat{\underset{\sim}{A}}^{(2)} + \widehat{\underset{\sim}{p}}^{(2)}\widehat{\underset{\sim}{q}}^{(2)*}$ is lower Hessenberg, $\widehat{\lambda}_2 = \widehat{\underset{\sim}{A}}^{(2)}_{1,1} + \widehat{\underset{\sim}{p}}^{(2)}_1\widehat{\underset{\sim}{q}}^{(2)*}_1$ is an eigenvalue of $\widehat{\underset{\sim}{A}}^{(2)} + \widehat{\underset{\sim}{p}}^{(2)}\widehat{\underset{\sim}{q}}^{(2)*}$. Now define the unitary matrix $U^{(2)} \in \mathbb{C}^{n\times n}$ by the formula

$$U^{(2)} = \left[\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \underset{\sim}{U}^{(2)} & \\ 0 & & & \end{array}\right], \tag{167}$$

the matrix $\delta\widehat{A}^{(1)} \in \mathbb{C}^{n\times n}$ by

$$\delta\widehat{A}^{(1)} = \left[\begin{array}{c|ccc} 0 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & \delta\widehat{\underset{\sim}{A}}^{(1)} & \\ 0 & & & \end{array}\right], \tag{168}$$

and the vectors $\delta\widehat{p}^{(1)}, \delta\widehat{q}^{(1)} \in \mathbb{C}^n$, by

$$\delta\widehat{p}^{(1)} = \left[\begin{array}{c} 0 \\ \hline \delta\widehat{\underset{\sim}{p}}^{(1)} \end{array}\right], \tag{169}$$

and

$$\delta\widehat{q}^{(1)} = \left[\begin{array}{c} 0 \\ \hline \delta\widehat{\underset{\sim}{q}}^{(1)} \end{array}\right]. \tag{170}$$

Clearly, $\|\delta A^{(1)}\| \lesssim \|A\|u$, $\|\delta\widehat{p}^{(1)}\| \lesssim \|p\|u$, and $\|\delta\widehat{q}^{(1)}\| \lesssim \|q\|u$. Let $\widehat{A}^{(2)} \in \mathbb{C}^{n\times n}$ be defined by

$$\widehat{A}^{(2)} = U^{(2)}(\widehat{A}^{(1)} + \delta A^{(1)})U^{(2)*}, \tag{171}$$

and $\widehat{p}^{(2)}, \widehat{q}^{(2)} \in \mathbb{C}^n$ by

$$\widehat{p}^{(2)} = U^{(2)}(\widehat{p}^{(1)} + \delta\widehat{p}^{(1)}), \tag{172}$$

and

$$\widehat{q}^{(2)} = U^{(2)}(\widehat{q}^{(1)} + \delta\widehat{q}^{(1)}). \tag{173}$$

We first notice that $\widehat{A}^{(2)}_{1,\ell} + \widehat{p}^{(2)}_1 \widehat{q}^{(2)*}_\ell = 0$ for $\ell = 2, 3, \ldots, n$. Next, we observe that $\widehat{A}^{(2)}_{1,1} + \widehat{p}^{(2)}_1 \widehat{q}^{(2)*}_1 = \widehat{A}^{(1)}_{1,1} + \widehat{p}^{(1)}_1 \widehat{q}^{(1)*}_1 = \widehat{\lambda}_1$; therefore, $\widehat{\lambda}_1$ is an eigenvalue of $\widehat{A}^{(2)} + \widehat{p}^{(2)}\widehat{q}^{(2)*}$. We then observe that $(\widehat{A}^{(2)} + \widehat{p}^{(2)}\widehat{q}^{(2)*})_{2:n,2:n} = \widehat{\underset{\sim}{A}}^{(2)} + \widehat{\underset{\sim}{p}}^{(2)}\widehat{\underset{\sim}{q}}^{(2)*}$; therefore, $\widehat{\lambda}_2$ is an eigenvalue of $\widehat{A}^{(2)} + \widehat{p}^{(2)}\widehat{q}^{(2)*}$. Finally, letting $U = U^{(2)}U^{(1)}$ and substituting (154), (155), and (159) into (171)–(173) and expanding, it is straightforward to show that there exist matrices $\delta A \in \mathbb{C}^{n\times n}$ and vectors $\delta p, \delta q \in \mathbb{C}^n$ such that

$$\widehat{A}^{(2)} + \widehat{p}^{(2)}\widehat{q}^{(2)*} = U(A + \delta A)U^* + U(p + \delta p)(q + \delta q)^* U^*, \tag{174}$$

where $\|\delta A\| \lesssim \|A\|u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$. Therefore, $\widehat{\lambda}_1$ and $\widehat{\lambda}_2$ are eigenvalues of the matrix

$$(A + \delta A) + (p + \delta p)(q + \delta q), \tag{175}$$

where $\|\delta A\| \lesssim \|A\|u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$. The same proof can be repeated inductively to show this for all $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$.

■

The following theorem states that our explicit shifted QR algorithm is componentwise backward stable, for those eigenvalues for which the shifts are small.

**Theorem 4.7** (Explicit shifted QR)**.** *Suppose that $A \in \mathbb{C}^{n\times n}$ is a Hermitian matrix, that $p, q \in \mathbb{C}^n$, and that $A + pq^*$ is lower Hessenberg. Let $d$ and $\beta$ denote the diagonal and superdiagonal of $A$, respectively. Suppose that Algorithm 4 is carried out in floating point arithmetic with $\epsilon \lesssim \|A\|u$, and suppose that $\mu^{(\ell)}$ is the largest total shift encountered at any point during the course of the algorithm from $i = 1, 2, \ldots, \ell$ in the outer loop. Let $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_n$ denote the outputs of the algorithm. Then, for each $\ell = 1, 2, \ldots, n$, there exist a matrix $\delta A \in \mathbb{C}^{n\times n}$ and vectors $\delta p, \delta q \in \mathbb{C}^n$ such that $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_\ell$ are exact eigenvalues of the matrix*

$$(A + \delta A) + (p + \delta p)(q + \delta q)^*, \tag{176}$$

*where $\|\delta A\| \lesssim (\|A\| + |\mu^{(\ell)}|)u$, $\|\delta p\| \lesssim \|p\|u$, and $\|\delta q\| \lesssim \|q\|u$.*

**Proof.** The proof is essentially identical to the proof of Theorem 4.6, and we omit it. ■

**Remark 4.1.** Notice that Theorems 4.6 and 4.7 do not make any mention of convergence. What they say is that, *if* the algorithm converges, then it is componentwise backward stable. We observe that, in practice, Algorithm 4 always converges rapidly, at least quadratically, for $\epsilon \approx \|A\|u$.

**Remark 4.2.** Notice that the bound on $\delta A$ in Theorem 4.7 involves $\mu^{(\ell)}$, which is the largest total shift encountered at any point during the calculation of $\widehat{\lambda}_1, \widehat{\lambda}_2, \ldots, \widehat{\lambda}_\ell$. While this bound appears weaker than the corresponding bound in Theorem 4.6, in practice it

turns out to be essentially the same, as follows. We can always assume that $A$ is much smaller than $p$, or $q$, or both; if this isn't the case, then componentwise stability no longer has any special meaning, since it follows immediately from the usual Bauer-Fike perturbation bounds (see [7]). Furthermore, we tend to be interested in the componentwise backward stability of small eigenvalues $\widehat{\lambda}_i$, where $|\widehat{\lambda}_i| \approx \|A\|$. If we perform a few iterations of *unshifted* QR on the matrix, then the eigenvalues of the top-left $2\times 2$ block will approach the two smallest eigenvalues of the matrix (recalling that our QR algorithm works with lower Hessenberg matrices). If we now use Algorithm 4, we'll find that the initial shift is small and, as a result, all the eigenvalues are computed roughly in order from smallest to largest. This means that $|\mu^{(i)}| \approx |\widehat{\lambda}_i|$ and (approximately) $\widehat{\lambda}_1 < \widehat{\lambda}_2 < \cdots < \widehat{\lambda}_i$. For $\widehat{\lambda}_i$ such that $|\widehat{\lambda}_i| \approx \|A\|$, we have then that the bound $\|\delta A\| \lesssim (\|A\| + |\mu^{(i)}|)u$ becomes $\|\delta A\| \lesssim \|A\|u$. Finally, we point out that the dependence of the bound on $\mu^{(\ell)}$ could likely be removed entirely by reformulating our QR algorithm as an implicit method.

# 5 Numerical Results

In this section, we demonstrate the componentwise backward stability of our shifted QR algorithm (see Algorithm 4) by illustrating its stability when it is used as a rootfinding algorithm (see Sections 2.3 and 2.4). Consider a polynomial $p(x)$ of order $n$, not necessarily monic, expressed in a Chebyshev polynomial basis

$$p(x) = \sum_{j=0}^{n} a_j T_j(x), \tag{177}$$

where $a_j \in \mathbb{R}$ and $T_j(x)$ is the Chebyshev polynomial of order $j$. By Theorem 2.7 and Remark 2.1, we have that if the eigenvalues of the linearization (17), where $c_j = a_j/a_n$, $j = 0, 1, \ldots, n$, are computed by a componentwise backward stable algorithm, then the computed roots $\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_n$ are the exact roots of the perturbed polynomial

$$p(x) + \delta p(x) = \sum_{j=0}^{n} (a_j + \delta a_j) T_j(x), \tag{178}$$

where

$$\frac{\|\delta a\|}{\|a\|} \lesssim u. \tag{179}$$

By applying our QR algorithm to linearizations of various polynomials $p(x)$, we demonstrate our algorithm's componentwise backward stability by showing that the bound (179) always holds.

We estimate the size of the backward error $\delta a$ in the coefficients by using the following observation (see the discussion accompanying Table 1 in [26]). By the definition of $p(x) + \delta p(x)$, we have that $(p + \delta p)(\widehat{x}_i) = 0$ for $i = 1, 2, \ldots, n$. From (177) and (178), it follows that

$$p(\widehat{x}_i) = p(\widehat{x}_i) - (p + \delta p)(\widehat{x}_i) = -\sum_{j=0}^{n} \delta a_j T_j(\widehat{x}_j). \tag{180}$$

Since $-1 \le T_j(x) \le 1$ for all $j$ when $x \in [-1,1]$, we have

$$p(\widehat{x}_i) \approx \|\delta a\|, \tag{181}$$

whenever $\widehat{x}_i \in \mathbb{C}$ is not too far from the interval $[-1,1]$.

Even though $\widehat{x}_i$ is already a floating point number, the value $p(\widehat{x}_i)$ cannot be computed exactly in floating point arithmetic. Letting $\widehat{p}(\widehat{x}_i)$ denote the approximation to $p(\widehat{x}_i)$ computed in floating point arithmetic, we know that

$$\widehat{p}(\widehat{x}_i) \approx p(\widehat{x}_i) + \kappa(p;\widehat{x}_i)u, \tag{182}$$

where

$$\kappa(p;\widehat{x}_i) = |\widehat{x}_i||p'(\widehat{x}_i)| \tag{183}$$

is the absolute condition number of $p(x)$ at $x = \widehat{x}_i$. When $\kappa(p;\widehat{x}_i)$ is large, the error in evaluating $p(\widehat{x}_i)$ dominates, while when $\kappa(p;\widehat{x}_i)$ is of modest size, we have $\widehat{p}(\widehat{x}_i) \approx p(\widehat{x}_i)$. In this section, we investigate the quantity

$$\eta(p;\widehat{x}_i) = \frac{\widehat{p}(\widehat{x}_i)}{\max(\kappa(p;\widehat{x}_i), \|a\|)}, \tag{184}$$

for various polynomials $p(x)$. When $\kappa(p;\widehat{x}_i) \ge \|a\|$, we have

$$|\eta(p;\widehat{x}_i)| = \frac{|\widehat{p}(\widehat{x}_i)|}{\kappa(p;\widehat{x}_i)} \approx \frac{|p(\widehat{x}_i)|}{\kappa(p;\widehat{x}_i)} + u \le \frac{|p(\widehat{x}_i)|}{\|a\|} + u. \tag{185}$$

When $\kappa(p;\widehat{x}_i) \le \|a\|$,

$$|\eta(p;\widehat{x}_i)| = \frac{|\widehat{p}(\widehat{x}_i)|}{\|a\|} \approx \frac{|p(\widehat{x}_i)|}{\|a\|} + \frac{\kappa(p;\widehat{x}_i)}{\|a\|}u \le \frac{|p(\widehat{x}_i)|}{\|a\|} + u. \tag{186}$$

Thus, if our QR algorithm is indeed componentwise backward stable and (179) is satisfied, then, by (181), we expect to find that

$$\eta(p;\widehat{x}_i) \approx u, \tag{187}$$

for all polynomials $p(x)$.

For $p(\widehat{x}_i)$ to be a good approximation to $\|\delta a\|$ (see (181)), we stated that $\widehat{x}_i \in \mathbb{C}$ should be "not too far from the interval $[-1,1]$." We make this notion precise as follows. Let $z_i$, $i = 1, 2, \ldots, n$ denote the exact roots of the order-$n$ polynomial $p(x)$, and let $\widehat{z}_i$ denote the computed roots. We select roots close to the interval $[-1,1]$ by choosing some $\delta > 0$ (for example, $\delta = 10^{-3}$), and letting $\widehat{x}_i \in \mathbb{R}$ denote the real part of all roots $\widehat{z}_i$ that are inside the rectangle

$$\{z \in \mathbb{C} : 1 - \delta < \mathrm{Re}(z) < 1 + \delta, -\delta < \mathrm{Im}(z) < \delta\}. \tag{188}$$

If the polynomial $p(x)$ has the real root $z_i$, then taking the real part of $\widehat{z}_i$ will not result in any additional error. The number of real roots inside the region (188) will often be less than the order $n$, and we denote the number of such roots by $n_{\mathrm{roots}}$.

In our numerical experiments, we compute the eigenvalues of the colleague matrix using three different algorithms: our Algorithm 4; MATLAB's `eig` function; and MATLAB's `eig` function with balancing turned off (using the option `'nobalance'`), which we call `eig_nb`. For our experiments in extended (quadruple) precision, we use the Advanpix Multiprecision Computing Toolbox and its implementation of `eig` (see [1]). Since the Advanpix Multiprecision Computing Toolbox's `eig` function always balances the matrix, and does not support the `'nobalance'` option, we omit the test of `eig_nb` in extended precision.

For each example, we report the degree of the underlying polynomial, the order $n$ of the Chebyshev expansion used to approximate it, the size of the vector $c$ in the Euclidean norm, the Frobenius norm of the completely balanced colleague matrix, which we denote by $\texttt{bal}(C)$, the number $n_{\text{roots}}$ of computed roots inside the region (188) for the given value of $\delta > 0$, the size $\max_i |z_i|$ of the largest complex root of the colleague matrix, and the value of $\max_i |\eta(p; \widehat{x}_i)|$, where the maximum is taken over all of the real parts of the computed roots inside the region (188).

We implemented our algorithm in FORTRAN 77, and compiled it using Lahey/Fujitsu Fortran 95 Express, Release L6.20e. For the timing experiments, the Fortran codes were compiled using the Intel Fortran Compiler, version 19.0.2.187, with the `-fast` flag. The MATLAB experiments were performed using MATLAB R2019b, version 9.7.0.1190202, and the extended precision MATLAB experiments were performed in quadruple precision (`mp.Digits(34)`) using the Advanpix Multiprecision Computing Toolbox, version 4.8.0, Build 14100. All experiments we conducted on a ThinkPad laptop, with 16GB of RAM and an Intel Core i7-8550U CPU.

### 5.1 $p_{\mathbf{rand}}(x)$: Polynomials with Random Coefficients

Following [14], we construct polynomials $p_{\text{rand}}(x)$ by sampling Chebyshev expansion coefficients $a_i$ independently from a standard normal distribution, so that $a_i \sim N(0, 1)$, for $i = 0, 1, \ldots, n-1$. Then, we choose the desired value of $\|c\|$ by setting $a_n = \|a\|/\|c\|$, so that the vector of coefficients $c$ appearing in the colleague matrix (17), where $c_i = a_i/a_n$ for $i = 1, 2, \ldots, n$, has the specified norm. For this example, we choose $n = 30$ and set $\delta = 10^{-5}$ to extract the real roots (see formula (188)).

We report the results in Figure 3. We see that our algorithm shows the expected backward stability over the entire range of $\|c\|$, while MATLAB's `eig`, both balanced and unbalanced, shows the expected growth with $\|c\|$ (see the discussion in Section 2.4). Interestingly, for this example, balancing appears to only improve the error by an order of magnitude or two, while leaving the growth in the error with respect to $\|c\|$ unchanged. This turns out be completely consistent the following explanation. Balancing the colleague matrix can reduce the magnitude of the all elements from $\|c\|$ to $\|c\|^{\frac{1}{2}}$, except for the element in the $(n, n)$-position, which balancing cannot change. In this example, all of the elements of the vector $c$ are around the same size as $\|c\|$, so there are $n$ large elements in the colleague matrix. Thus, balancing reduces the number of large elements of size $\|c\|$ from $n$ to 1, resulting in an $n$-fold reduction in the norm of the matrix. In this example, $n = 30$, which corresponds well with the approximately 30-fold reduction in error due to balancing that we observe in Figure 3.

We found that the colleague matrix has a single large eigenvalue of the size $\|c\|$, and the rest of the eigenvalues are small. This is not surprising, since there is an entry of size

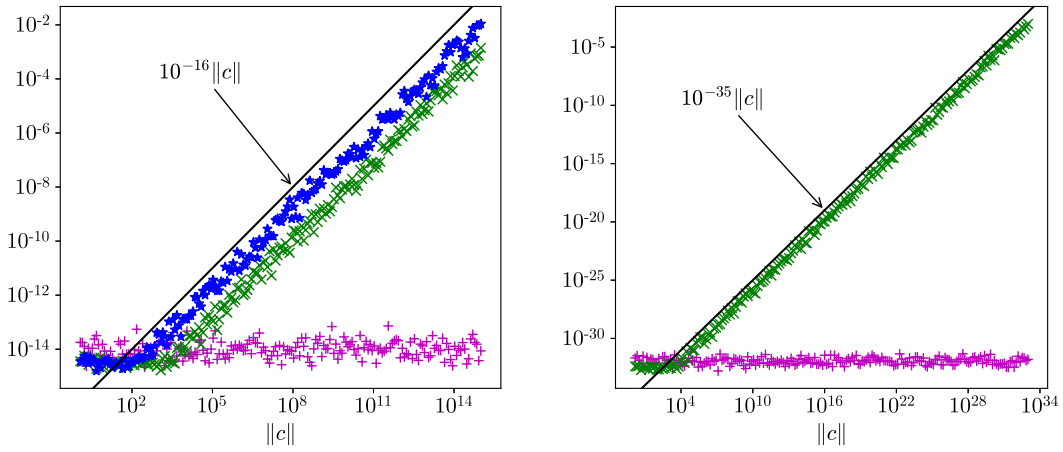


Figure 3: The values of $\max_i |\eta(p;\widehat{x}_i)|$ for various values of $\|c\|$, in double precision (left) and quadruple precision (right), for the polynomials $p_{\text{rand}}(x)$ of order $n = 30$, computed by our algorithm, `eig`, and `eig_nb`, with $\delta = 10^{-5}$. The values are indicated for our algorithm with purple crosses (+), for `eig` with green x's (×), and for `eig_nb` with blue stars ($\star$).

$\|c\|$ is the $(n, n)$-position of the matrix (from which it follows that $Ce_n \approx \|c\|e_n$). Thus, for all three algorithms, $\max_i |\widehat{z}_i| \approx \|c\|$.

### 5.2 $p_{\text{wilk}}(x)$: Wilkinson's Polynomial

Here we consider the famous Wilkinson polynomial, normalized so that all of its roots are inside the interval $[-1, 1]$, defined by the formula

$$p_{\text{wilk}}(x) = \prod_{i=1}^{m} \Big( x - \Big( \frac{2i}{m+1} - 1 \Big) \Big). \tag{189}$$

We construct an order-$n$ Chebyshev expansion of this degree-$m$ polynomial, sampling it at $n$ Chebshev points and applying a linear transformation to obtain the expansion coefficients (see [36]). We then compute the eigenvalues of the colleague matrix, and set $\delta = 10^{-3}$ to extract the real roots (see formula (188)). The results of our numerical experiment are shown in Tables 1 and 2. We observe that our algorithm is backward stable, while `eig` loses accuracy whenever the roots of the colleague matrix are large. Plots of the real and complex roots of the order-100 Chebyshev expansion are shown for various degrees of $p_{\text{wilk}}(x)$ in Figure 4. When the order of the Wilkinson polynomial becomes large, spurious real roots begin appearing in the middle of the interval $[-1, 1]$. It turns out that the roots computed by our algorithm are still backward stable, even in this situation; the function is so small near the middle of the interval that a small relative perturbation in the Chebyshev coefficients causes additional roots to appear.

**Remark 5.1.** The remarkable stability of `eig` for many of the examples in Tables 1 and 2 is explained by the following observation. The colleague matrix is the sum of a tridiagonal matrix and a matrix that is all zeros except for the last row, which is essentially equal

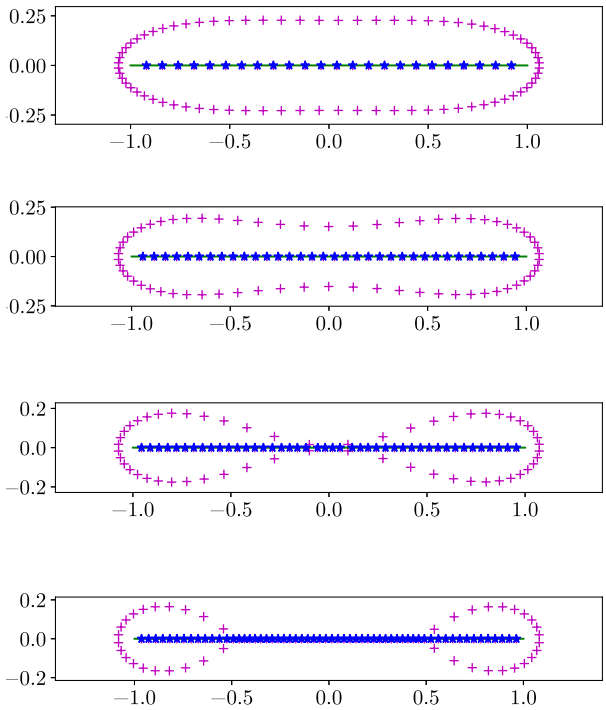


Figure 4: The roots of the Chebyshev expansion of order 100 of the Wilkinson polynomial $p_{\text{wilk}}(x)$, of various degrees, computed by our algorithm. The complex roots $\widehat{z}_i$ are plotted with purple crosses (+) and the real roots $\widehat{x}_i$ are plotted with blue stars ($\star$). The Wilkinson polynomial has, in order from top to bottom, orders 24, 34, 44, and 54. Observe that the spurious complex roots are well-separated from the interval $[-1, 1]$ when the order is low, but eventually meet the interval when the order is large.

to the coefficient vector $c$ (see formula (17)). When there are large elements of $c$ near the tail of the vector, the corresponding large entries in the colleague matrix cannot be balanced away, since they are very close to the diagonal of the matrix. On the other hand, when all of the elements of $c$ near the tail of the vector are relatively small, and the large elements of $c$ appear near the head of the vector, these large elements can be easily balanced away, since they are far from the diagonal, and the corresponding elements on the other side of the diagonal are all zero. The coefficient vector $c$ is usually large only because the last coefficient of the corresponding non-monic Chebyshev expansion is small. If the function being approximated by this non-monic Chebyshev expansion has been adequately represented, then taking additional terms in the expansion will result in corresponding expansion coefficients which are all machine epsilon in size. Thus, adding terms to the Chebyshev expansion has the effect of adding elements of size approximately one to the tail of the coefficient vector $c$; if enough such elements are added, then all the large elements of $c$ will be closer to the head of the vector, and can be balanced away. We also observe that, not unexpectedly, the size of the largest eigenvalue of the colleague matrix is approximately the same size as the norm of the colleague matrix after balancing. Thus, if enough terms are taken in a Chebyshev expansion, all of the

| | | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|
| Degree | $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 14 | 100 | $0.45 \cdot 10^{14}$ | $0.32 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 14 | $0.16 \cdot 10^{-13}$ | 14 | $0.71 \cdot 10^{-14}$ |
| 24 | 24 | $0.95 \cdot 10^{4}$ | $0.65 \cdot 10^{1}$ | $0.92 \cdot 10^{0}$ | 24 | $0.15 \cdot 10^{-14}$ | 24 | $0.32 \cdot 10^{-14}$ |
| | 25 | $0.66 \cdot 10^{15}$ | $0.35 \cdot 10^{11}$ | $0.35 \cdot 10^{11}$ | $14^{\dagger}$ | $0.11 \cdot 10^{-4}$ | 24 | $0.19 \cdot 10^{-14}$ |
| | 26 | $0.22 \cdot 10^{15}$ | $0.11 \cdot 10^{6}$ | $0.76 \cdot 10^{5}$ | 24 | $0.51 \cdot 10^{-10}$ | 24 | $0.24 \cdot 10^{-14}$ |
| | 27 | $0.40 \cdot 10^{16}$ | $0.66 \cdot 10^{4}$ | $0.38 \cdot 10^{4}$ | 24 | $0.90 \cdot 10^{-12}$ | 24 | $0.19 \cdot 10^{-14}$ |
| | 28 | $0.12 \cdot 10^{15}$ | $0.37 \cdot 10^{3}$ | $0.17 \cdot 10^{3}$ | 24 | $0.58 \cdot 10^{-13}$ | 24 | $0.14 \cdot 10^{-14}$ |
| | 100 | $0.38 \cdot 10^{14}$ | $0.24 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 24 | $0.10 \cdot 10^{-13}$ | 24 | $0.24 \cdot 10^{-14}$ |
| 34 | 100 | $0.34 \cdot 10^{14}$ | $0.17 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 34 | $0.14 \cdot 10^{-13}$ | 34 | $0.12 \cdot 10^{-13}$ |
| 44 | 100 | $0.32 \cdot 10^{14}$ | $0.16 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 44 | $0.53 \cdot 10^{-13}$ | 44 | $0.41 \cdot 10^{-14}$ |
| 54 | 100 | $0.30 \cdot 10^{14}$ | $0.16 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 60 | $0.15 \cdot 10^{-13}$ | 60 | $0.28 \cdot 10^{-13}$ |

Table 1: The results of computing the roots of the Wilkinson polynomial $p_{\text{wilk}}(x)$, using our algorithm and `eig`, with $\delta = 10^{-3}$. $^{\dagger}$The error was so large here that some roots were outside of the region (188).

| | | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|
| Degree | $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 14 | 100 | $0.12 \cdot 10^{34}$ | $0.38 \cdot 10^{3}$ | $0.14 \cdot 10^{1}$ | 14 | $0.22 \cdot 10^{-31}$ | 14 | $0.15 \cdot 10^{-32}$ |
| 24 | 24 | $0.95 \cdot 10^{4}$ | $0.65 \cdot 10^{1}$ | $0.92 \cdot 10^{0}$ | 24 | $0.44 \cdot 10^{-32}$ | 24 | $0.66 \cdot 10^{-33}$ |
| | 25 | $0.95 \cdot 10^{33}$ | $0.50 \cdot 10^{29}$ | $0.50 \cdot 10^{29}$ | $12^{\dagger}$ | $0.29 \cdot 10^{-4}$ | 24 | $0.78 \cdot 10^{-32}$ |
| | 26 | $0.16 \cdot 10^{33}$ | $0.94 \cdot 10^{14}$ | $0.66 \cdot 10^{14}$ | 24 | $0.11 \cdot 10^{-19}$ | 24 | $0.18 \cdot 10^{-32}$ |
| | 27 | $0.11 \cdot 10^{36}$ | $0.22 \cdot 10^{11}$ | $0.11 \cdot 10^{11}$ | 24 | $0.55 \cdot 10^{-23}$ | 24 | $0.76 \cdot 10^{-32}$ |
| | 28 | $0.22 \cdot 10^{33}$ | $0.13 \cdot 10^{8}$ | $0.62 \cdot 10^{7}$ | 24 | $0.34 \cdot 10^{-27}$ | 24 | $0.28 \cdot 10^{-32}$ |
| | 100 | $0.94 \cdot 10^{33}$ | $0.29 \cdot 10^{3}$ | $0.14 \cdot 10^{1}$ | 24 | $0.53 \cdot 10^{-31}$ | 24 | $0.37 \cdot 10^{-32}$ |
| 34 | 100 | $0.70 \cdot 10^{33}$ | $0.26 \cdot 10^{3}$ | $0.15 \cdot 10^{1}$ | 34 | $0.21 \cdot 10^{-31}$ | 34 | $0.68 \cdot 10^{-32}$ |
| 44 | 100 | $0.53 \cdot 10^{33}$ | $0.23 \cdot 10^{3}$ | $0.16 \cdot 10^{1}$ | 44 | $0.54 \cdot 10^{-32}$ | 44 | $0.40 \cdot 10^{-32}$ |
| 54 | 100 | $0.42 \cdot 10^{33}$ | $0.16 \cdot 10^{3}$ | $0.18 \cdot 10^{1}$ | 54 | $0.28 \cdot 10^{-31}$ | 54 | $0.73 \cdot 10^{-32}$ |

Table 2: The results of computing the roots of the Wilkinson polynomial $p_{\text{wilk}}(x)$ in extended precision, using our algorithm and `eig`, with $\delta = 10^{-3}$. $^{\dagger}$The error was so large here that some roots were outside of the region (188).

eigenvalues of the colleague matrix will eventually be small. All of this indicates that, provided enough terms are taken, a dense eigensolver combined with balancing can result in a backward stable rootfinding algorithm. Of course, we note that balancing the colleague matrix destroys the Hermitian plus rank-1 structure, which bars the use of structured QR algorithms depending on this property.

### 5.3 $f_{\text{sin}}(x)$: A Smooth Function

Here we construct an order-$n$ Chebyshev expansion of the smooth function

$$f_{\text{sin}}(x) = \sin(2 + 20(x + 0.222)^2). \tag{190}$$

Since $f_{\text{sin}}(x)$ is analytic, its expansion coefficients decay exponentially. When $i \geq 80$, the coefficients $a_i$ are around $10^{-14}$ in size (see Figure 5). If the coefficients are computed in extended precision, then, when $i \geq 125$, the coefficients $a_i$ are around $10^{-34}$ in size. Since the function is approximated accurately by a Chebyshev expansion, its roots can

be computed from the corresponding colleague matrix (see, for example, [12] for a nice discussion). The results of our numerical experiment are shown in Tables 3 and 4. Plots of the real and complex roots of the order-100 Chebyshev expansion are shown in Figure 6.

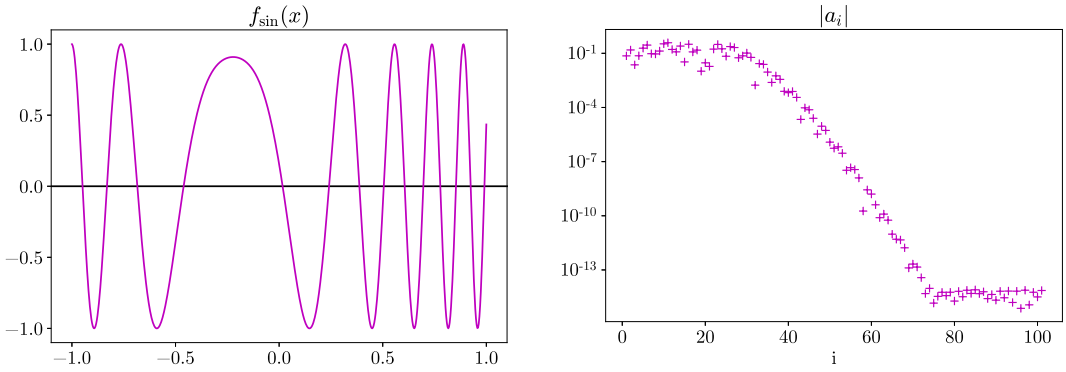


Figure 5: The function $f_{\sin}(x)$, shown on the left, and the magnitude of its Chebyshev expansion coefficients $a_i$, shown on the right.

| | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|
| $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 80 | $0.89 \cdot 10^{15}$ | $0.16 \cdot 10^{2}$ | $0.29 \cdot 10^{1}$ | 14 | $0.74 \cdot 10^{-14}$ | 14 | $0.10 \cdot 10^{-13}$ |
| 100 | $0.14 \cdot 10^{15}$ | $0.14 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 14 | $0.84 \cdot 10^{-14}$ | 14 | $0.26 \cdot 10^{-13}$ |

Table 3: The results of computing the roots of the Chebyshev expansion of the function $f_{\sin}(x)$, using our algorithm and `eig`, with $\delta = 10^{-3}$.

| | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|
| $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 125 | $0.92 \cdot 10^{33}$ | $0.30 \cdot 10^{2}$ | $0.25 \cdot 10^{1}$ | 14 | $0.53 \cdot 10^{-32}$ | 14 | $0.50 \cdot 10^{-31}$ |
| 200 | $0.49 \cdot 10^{32}$ | $0.29 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 14 | $0.12 \cdot 10^{-31}$ | 14 | $0.60 \cdot 10^{-31}$ |

Table 4: The results of computing the roots of the Chebyshev expansion of the function $f_{\sin}(x)$ in extended precision, using our algorithm and `eig`, with $\delta = 10^{-3}$.

### 5.4 $p_{\textbf{mult}}(x)$: A Polynomial with Multiple Roots

Here we construct an order $n$ Chebyshev expansion of the degree $m$ polynomial

$$f_{\text{mult}}(x) = (x + \tfrac{1}{2})(x + \tfrac{1}{3})(x + 0.61)(x - 0.121) \prod_{i=1}^{m-4} (x - (1 - 10^{-3})). \tag{191}$$

This polynomial has four simple roots on the interval $[-1, 1]$, and a root of multiplicity $(m-4)$ at the point $1 - 10^{-3}$ (see Figure 7). The results of our numerical experiments are shown in Tables 5 and 6. We observe that, in double precision, when the multiplicity of the root is greater than or equal to 5, not all real roots are found. This is because

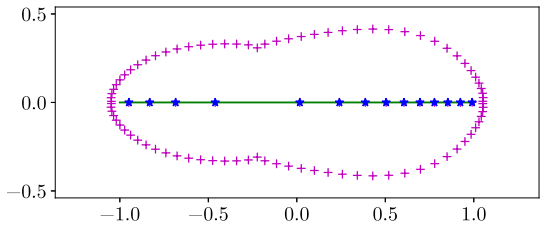


Figure 6: The roots of the Chebyshev expansion of order 100 of the function $f_{\sin}(x)$. The complex roots $\widehat{z}_i$ are plotted with purple crosses (+) and the real roots $\widehat{x}_i$ are plotted with blue stars ($\star$).

the error in these roots is approximately equal to $\epsilon^{\frac{1}{5}}$, and when $\epsilon \approx 10^{-14}$, we have that $\epsilon^{\frac{1}{5}} \approx 1.6 \cdot 10^{-3}$; when $\delta = 10^{-3}$, this means that some of these roots can be outside the region (188). Likewise, since when $\epsilon \approx 10^{-34}$, $\epsilon^{\frac{1}{12}} \approx 1.4 \cdot 10^{-3}$, it follows that in extended precision, real roots are missed when their multiplicity is greater than or equal to approximately 12. See the excellent discussion in [13] for more details.

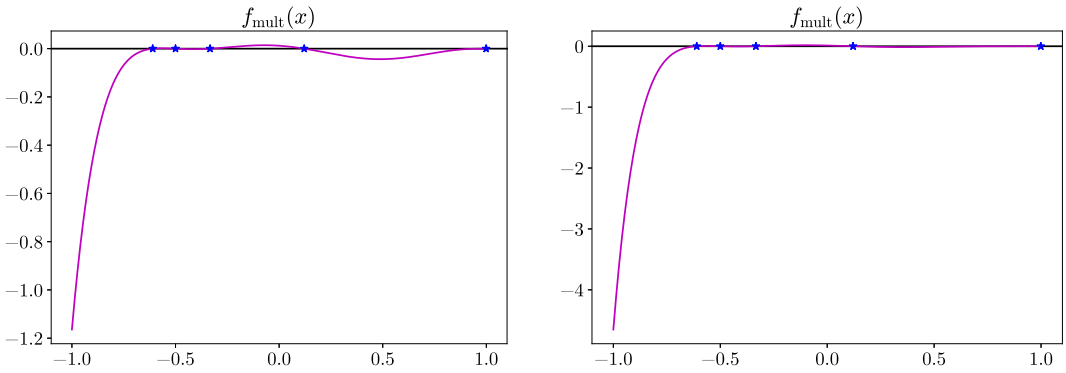


Figure 7: The polynomial $p_{\text{mult}}(x)$ of order 7 on the left, and order 9 on the right. The roots are indicated with blue stars ($\star$).

### 5.5 $p_{\text{yuji}}(x)$: A Pathological Example from [26]

Here we consider the order-8 polynomial

$$p_{\text{yuji}}(x) = \sum_{i=0}^{8} a_i T_i(x), \tag{192}$$

where the coefficient vector $a$ is given by

$$a = \left( \; -\tfrac{1}{10} \quad -\tfrac{1}{10} \quad -\tfrac{1}{10} \quad -\tfrac{1}{10} \quad -\tfrac{1}{10} \quad -\tfrac{1}{10} \quad 10^{-10} \quad 1 \quad 10^{-15} \; \right), \tag{193}$$

described in §6.1 of [26] (in [26], the authors set the last element of the coefficient vector to $10^{-20}$; we set it close to machine epsilon instead). Observe that the entry in the

| | | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|
| Degree | $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 7 | 100 | $0.10 \cdot 10^{15}$ | $0.38 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 7 | $0.82 \cdot 10^{-14}$ | 7 | $0.14 \cdot 10^{-14}$ |
| 8 | 8 | $0.12 \cdot 10^{3}$ | $0.50 \cdot 10^{1}$ | $0.10 \cdot 10^{1}$ | 8 | $0.59 \cdot 10^{-15}$ | 8 | $0.93 \cdot 10^{-15}$ |
| | 9 | $0.54 \cdot 10^{15}$ | $0.22 \cdot 10^{13}$ | $0.22 \cdot 10^{13}$ | $5^{\dagger}$ | $0.33 \cdot 10^{-4}$ | 8 | $0.11 \cdot 10^{-14}$ |
| | 10 | $0.61 \cdot 10^{15}$ | $0.16 \cdot 10^{7}$ | $0.11 \cdot 10^{7}$ | $6^{\dagger}$ | $0.14 \cdot 10^{-9}$ | 8 | $0.88 \cdot 10^{-15}$ |
| | 11 | $0.79 \cdot 10^{15}$ | $0.17 \cdot 10^{5}$ | $0.93 \cdot 10^{4}$ | 8 | $0.20 \cdot 10^{-11}$ | 8 | $0.83 \cdot 10^{-15}$ |
| | 100 | $0.99 \cdot 10^{14}$ | $0.32 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | 8 | $0.64 \cdot 10^{-14}$ | 8 | $0.26 \cdot 10^{-15}$ |
| 9 | 100 | $0.97 \cdot 10^{14}$ | $0.32 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | $8^{\diamond}$ | $0.99 \cdot 10^{-14}$ | $8^{\diamond}$ | $0.88 \cdot 10^{-14}$ |
| 10 | 100 | $0.96 \cdot 10^{14}$ | $0.26 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | $8^{\diamond}$ | $0.73 \cdot 10^{-15}$ | $8^{\diamond}$ | $0.38 \cdot 10^{-15}$ |
| 13 | 100 | $0.92 \cdot 10^{14}$ | $0.22 \cdot 10^{2}$ | $0.11 \cdot 10^{1}$ | $12^{\diamond}$ | $0.12 \cdot 10^{-14}$ | $12^{\diamond}$ | $0.88 \cdot 10^{-15}$ |

Table 5: The results of computing the roots of the polynomial $p_{\text{mult}}(x)$, using our algorithm and `eig`, with $\delta = 10^{-3}$. $^{\dagger}$The error was so large here that some roots were outside of the region (188). $^{\diamond}$The multiplicity of the rightmost root was so large here that some roots were outside of the region (188).

| | | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|
| Degree | $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 10 | 100 | $0.69 \cdot 10^{33}$ | $0.29 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | 10 | $0.94 \cdot 10^{-32}$ | 10 | $0.51 \cdot 10^{-33}$ |
| 11 | 11 | $0.75 \cdot 10^{4}$ | $0.98 \cdot 10^{1}$ | $0.10 \cdot 10^{1}$ | 11 | $0.76 \cdot 10^{-33}$ | 11 | $0.12 \cdot 10^{-32}$ |
| | 12 | $0.16 \cdot 10^{34}$ | $0.11 \cdot 10^{30}$ | $0.11 \cdot 10^{30}$ | $6^{\dagger}$ | $0.28 \cdot 10^{-5}$ | 11 | $0.95 \cdot 10^{-33}$ |
| | 13 | $0.37 \cdot 10^{34}$ | $0.53 \cdot 10^{15}$ | $0.35 \cdot 10^{15}$ | $7^{\dagger}$ | $0.63 \cdot 10^{-19}$ | 11 | $0.66 \cdot 10^{-33}$ |
| | 14 | $0.25 \cdot 10^{34}$ | $0.68 \cdot 10^{10}$ | $0.35 \cdot 10^{10}$ | 11 | $0.51 \cdot 10^{-25}$ | 11 | $0.37 \cdot 10^{-33}$ |
| | 100 | $0.70 \cdot 10^{33}$ | $0.37 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | 11 | $0.11 \cdot 10^{-31}$ | 11 | $0.15 \cdot 10^{-32}$ |
| 12 | 100 | $0.71 \cdot 10^{33}$ | $0.29 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | 12 | $0.20 \cdot 10^{-31}$ | 12 | $0.81 \cdot 10^{-33}$ |
| 13 | 100 | $0.72 \cdot 10^{33}$ | $0.30 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | 13 | $0.17 \cdot 10^{-31}$ | 13 | $0.11 \cdot 10^{-32}$ |
| 14 | 100 | $0.72 \cdot 10^{33}$ | $0.28 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | 14 | $0.24 \cdot 10^{-31}$ | 14 | $0.12 \cdot 10^{-32}$ |
| 15 | 100 | $0.73 \cdot 10^{33}$ | $0.28 \cdot 10^{3}$ | $0.13 \cdot 10^{1}$ | $9^{\diamond}$ | $0.21 \cdot 10^{-31}$ | $11^{\diamond}$ | $0.21 \cdot 10^{-31}$ |

Table 6: The results of computing the roots of the polynomial $p_{\text{mult}}(x)$ in extended precision, using our algorithm and `eig`, with $\delta = 10^{-3}$. $^{\dagger}$The error was so large here that some roots were outside of the region (188). $^{\diamond}$The multiplicity of the rightmost root was so large here that some roots were outside of the region (188).

bottom right corner of the corresponding colleague matrix is around $10^{15}$ in size. This polynomial has seven real roots on the interval $[-1, 1]$, and a single large imaginary root. We report the results of our numerical experiment in Table 7. Clearly, `eig` struggles to produce any accuracy at all, while our algorithm returns all the roots to machine precision.

### 5.6 $f_{\textbf{cas}}(x)$: A Pathological Example from [14]

Here we consider the order-$n$ Chebyshev expansion of the smooth function

$$f_{\text{cas}}(x) = \sin\Big(\frac{1}{x^2 + 10^{-2}}\Big), \tag{194}$$

described in [14]. The first 1430 Chebyshev expansion coefficients of $f_{\text{cas}}(x)$ are shown in Figure 8. This function is highly oscillatory, and requires a Chebyshev expansion of order

| | | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|---|
| Degree | $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 8 | 8 | $0.10 \cdot 10^{16}$ | $0.50 \cdot 10^{15}$ | $0.50 \cdot 10^{15}$ | 7 | $0.21 \cdot 10^{-1}$ | 7 | $0.77 \cdot 10^{-14}$ |

Table 7: The results of computing the roots of the polynomial $p_{\text{yuji}}(x)$, using our algorithm and `eig`, with $\delta = 10^{-3}$.

at least 1430 to resolve it. Our numerical experiments are shown in Table 8. Plots of the real and complex roots of the order-1600 Chebyshev expansion are shown in Figure 9.

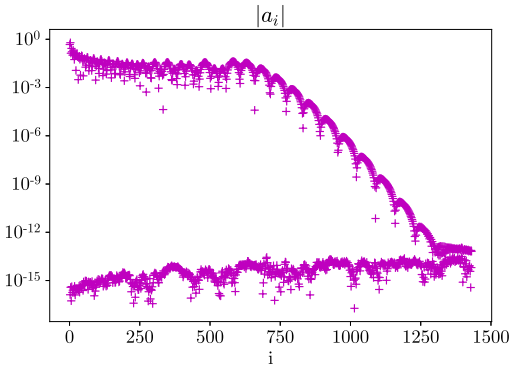


Figure 8: The magnitudes of the first 1430 Chebyshev expansion coefficients of $f_{\text{cas}}(x)$.

| | | | | `eig` | | Algorithm 4 | |
|---|---|---|---|---|---|---|---|
| $n$ | $\|c\|$ | $\|\texttt{bal}(C)\|$ | $\max_i \|z_i\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ | $n_{\text{roots}}$ | $\max_i \|\eta(p;\widehat{x}_i)\|$ |
| 1430 | $0.16 \cdot 10^{14}$ | $0.39 \cdot 10^{2}$ | $0.10 \cdot 10^{1}$ | 62 | $0.25 \cdot 10^{-13}$ | 62 | $0.98 \cdot 10^{-12}$ |

Table 8: The results of computing the roots of the Chebyshev expansion of the function $f_{\text{cas}}(x)$, using our algorithm and `eig`, with $\delta = 10^{-4}$.

### 5.7 CPU Times

The CPU times of our algorithm are compared to the times of MATLAB's `eig` in Figure 10. These timing experiments were performed on polynomials with random, independent, normally distributed Chebyshev expansion coefficients, with the last coefficient chosen so that the vector $c$ has the desired norm (see Section 5.1). We found that the CPU times do not depend on $\|c\|$, so we report the results only for $\|c\| = 2$. We observe that our algorithm is strictly faster than `eig`, even for small inputs, except perhaps for $n = 7$, for which our algorithm and `eig` cost about the same. The growth in CPU times taken by our algorithm agrees nicely with the expected asymptotic cost of $O(n^2)$, while `eig` shows a growth of $O(n^3)$.

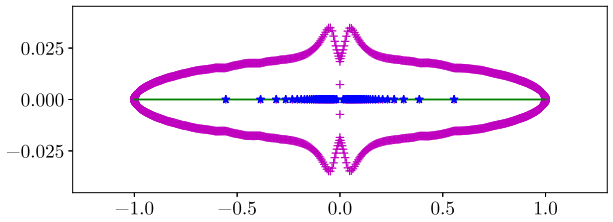


Figure 9: The roots of the Chebyshev expansion of order 1600 of the function $f_{\text{cas}}(x)$. The complex roots $\widehat{z}_i$ are plotted with purple crosses (+) and the real roots $\widehat{x}_i$ are plotted with blue stars ($\star$).

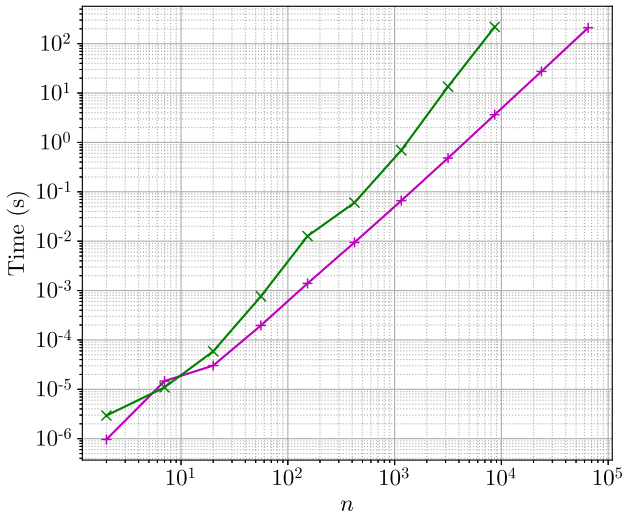


Figure 10: The CPU times of our algorithm, plotted with purple crosses (+), and the CPU times of `eig`, plotted with green x's ($\times$), for various values of $n$, where $n$ is the dimensionality of the colleague matrix.

## 6 Conclusions and Generalizations

In this report, we describe an explicit, $O(n^2)$ structured QR algorithm for colleague matrices (more generally, for Hessenberg matrices that are the sum of a Hermitian matrix and a rank-1 matrix), and prove that it is componentwise backward stable. These results can be generalized in several directions, of which we describe four. First, the algorithm can be modified in a fairly straightforward way to work on Hessenberg matrices that are the sum of a Hermitian matrix and a rank-$k$ perturbation (as opposed to a rank-1 perturbation). Like in the rank-1 case, most of the entries in the Hermitian part are inferred from the low rank part, except that they are inferred from a rank-$k$ matrix instead of a rank-1 matrix. The QR iteration proceeds similarly, the main difference being that the correction in Line 12 of Algorithm 1 becomes a correction to a row of an $n \times k$ matrix.

Second, the extension of this algorithm to an implicit, $O(n^2)$ structured QR algorithm

that is also componentwise backward stable is fairly straightforward. The key observation of this report (that, to maintain componentwise error bounds, a correction must be applied to the rank-1 part whenever an entry of the matrix is eliminated) can be applied to a bulge-chasing algorithm where the matrix is similarly represented by generators.

Third, we observe that this algorithm can be used to accelerate the calculation of eigenvalues of general matrices, not necessarily in Hessenberg form, that are representable as the sum of a Hermitian matrix and a rank-1 (or rank-$k$) matrix. Such matrices can be quickly reduced to Hessenberg form in $O(n^3)$ operations, and once they are in Hessenberg form, our $O(n^2)$ algorithm can be used to compute the eigenvalues. Thus, the cost of the algorithm is dominated by the reduction to Hessenberg form, which will have a much smaller constant than the standard algorithm for the evaluation of the eigenvalues of the original dense matrix. Furthermore, if the reduction to Hessenberg form can be done in a componentwise backward stable fashion, then this scheme results in a componentwise backward stable eigensolver for general matrices of the form Hermitian plus rank-1 (or rank-$k$).

Fourth, we observe that our algorithm can be used to find the roots of polynomials expressed in other bases besides Chebyshev polynomials. It was observed in [5] that, given any orthogonal polynomial basis that satisfies a three-term recurrence relation, and given a polynomial expressed in that basis, it is possible to construct an analogue of the colleague matrix from the expansion coefficients. This matrix is a Hessenberg matrix that is the sum of a (not necessarily symmetric) tridiagonal matrix and a rank-1 matrix; matrices of this form are called *comrade matrices*. For all classical orthogonal polynomials, the tridiagonal part can made symmetric by balancing, without making any entries of the matrix much larger or much smaller. Our algorithm can then be applied to this new matrix, which is a Hessenberg matrix that is the sum of a symmetric tridiagonal matrix and a rank-1 matrix.

## References


[1] Advanpix Multiprecision Computing Toolbox for MATLAB. Yokohama, Japan: Advanpix LLC. See `http://www.advanpix.com/`.

[2] Aurentz, J., T. Mach, L. Robol, R. Vandebril, D.S. Watkins. *Core-Chasing Algorithms for the Eigenvalue Problem.* SIAM, 2018.

[3] Aurentz, J.L., T. Mach, L. Robol, R. Vandelbril, and D.S. Watkins. "Fast and backward stable computation of roots of polynomials, part II: backward error analysis; companion matrix and companion pencil." *SIAM J. Matrix Anal. Appl.*, 39.3 (2018): 1245–1269.

[4] Aurentz, J.L., T. Mach, R. Vandelbril, and D.S. Watkins. "Fast and backward stable computation of roots of polynomials." *SIAM J. Matrix Anal. Appl.*, 36.3 (2015): 942–973.

[5] Barnett, S. "A companion matrix analogue for orthogonal polynomials." *Linear Algebra Appl.* 12 (1975): 197–208.

[6] Battles, Z. and L.N. Trefethen. "An extension of Matlab to continuous functions and operators." *SIAM J. Sci. Comput.* 25 (2004): 1743–1770.

[7] Bauer, F.L., C.T. Fike. "Norm and exclusion theorems." *Numer. Math.* 2 (1960): 137–141.

[8] Van Barel, M., R. Vandebril, P. Van Dooren, K. Frederix. "Implicit double shift QR-algorithm for companion matrices." *Numer. Math.* 116 (2010): 177–212.

[9] Bini, D.A., P. Bonito, Y. Eidelman, L. Gemignani, and I. Gohberg. "A fast implicit QR eigenvalue algorithm for companion matrices." *Linear Algebra Appl.* 432 (2010): 2006–2031.

[10] Bini, D.A., Y. Eidelman, L. Gemignani, and I. Gohberg. "Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices." *SIAM J. Matrix Anal. Appl.* 29.2 (2007): 566–585.

[11] Bini, D.A., L. Gemignani, V.Y. Pan. "Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equations." *Numer. Math.* 100 (2005): 373–408.

[12] Boyd, J.P. "Finding the Zeros of a Univariate Equation: Proxy Rootfinders, Chebyshev Interpolation, and the Companion Matrix." *SIAM Rev.* 55.2 (2013): 375–396.

[13] Boyd, J.P and D.H. Gally. "Numerical experiments on the accuracy of the Chebyshev-Frobenius companion matrix method for finding the zeros of a truncated series of Chebyshev polynomials." *J. Comput. Appl. Math.* 205 (2007): 281–295.

[14] Casulli, A. and L. Robol. "Rank-structured QR for Chebyshev rootfinding." arXiv:2010.11416v1 [math.NA], Oct. 2020.

[15] Chandrasekaran, S., M. Gu, J. Xia, J. Zhu. "A fast QR algorithm for companion matrices." *Recent Advances in Matrix and Operator Theory*, *Oper. Theory Adv. Appl.* 179 (2008): 111–143.

[16] Chebfun for MATLAB. See `http://www.chebfun.org/`.

[17] Corless, Robert M. "Generalized companion matrices in the Lagrange basis." *Proceedings EACA* (pp. 317–322). Santander, Spain: Universidad de Cantabria, 2004.

[18] Corless, R.M. Personal Communication. 13 July 2020.

[19] Edelman, A. and H. Murakami. "Polynomial roots from companion matrix eigenvalues." *Math. Comput.* 64.210 (1995): 763–776.

[20] Eidelman, Y., L. Gemignani, and I. Gohberg. "Efficient eigenvalue computation for quasiseparable Hermitian matrices under low rank perturbations." *Numer. Algor.* 47 (2008): 253–273.

[21] Good, I.J. "The colleague matrix, a Chebyshev analogue of the companion matrix." *Q. J. Math.*, 2.12 (1961): 61–68.

[22] Higham, N.J. *Accuracy and Stability of Numerical Algorithms.* 2nd ed. Philadelpha: SIAM, 2002.

[23] Lawrence, P.W., M. Van Barel, P. Van Dooren. "Backward error analysis of polynomial eigenvalue problems solved by linearization." *SIAM J. Matrix Anal. Appl.* 37.1 (2016): 123–144.

[24] McNamee, J.M. *Numerical Methods for Roots of Polynomials, Part I.* Elsevier, 2007.

[25] McNamee, J.M. and V.Y. Pan. *Numerical Methods for Roots of Polynomials, Part II.* Elsevier, 2013.

[26] Nakatsukasa, Y. and V. Noferini. "On the stability of computing polynomial roots via confederate linearizations." *Math. of Comput.*, 85.301 (2016): 2391–2425.

[27] Noferini, V., L. Robol, and R. Vandebril. "Structured backward errors in linearizations." arXiv:1912.04157v1 [math.NA], Dec. 2019.

[28] Pan, V.Y. "Solving a polynomial equation: some history and recent progress." *SIAM Rev.* 39.2 (1997): 187–220.

[29] Parlett, B.N. and C. Reinsch. "Balancing a matrix for calculation of eigenvalues and eigenvectors." *Numer. Math.* 13 (1963), 293–304.

[30] Pérez, J. and V. Noferini. "Chebyshev rootfinding via computing eigenvalues of colleague matrices: what is it stable?" *Math. Comput.* 86.306 (2017): 1741–1767.

[31] Peters, G. and J.H. Wilkinson. "Practical problems arising in the solution of polynomial equations." *J. Inst. Maths. Appl.* 8 (1971), 16–35.

[32] Serkh, K. "A fast, simple, and remarkably stable QR method for colleague matrices, and connections to event detection for the numerical solution of ODEs." *2020 SIAM/CAIMS: 2nd Joint Annual Meeting*, July, 2020. See `https://meetings.siam.org/sess/dsp_talk.cfm?p=106559`.

[33] Specht, W., "Die Lage der Nullstellen eines Polynoms III." *Math. Nach.* 16 (1957): 369–389.

[34] Specht, W. "Die Lage der Nullstellen eines Polynoms IV." *Math. Nach.* 21 (1960): 201–222.

[35] Tisseur, F. "Backward stability of the QR algorithm." *Equipe d'Analyse Numerique, Université Jean Monnet de Saint-Etienne Technical Report* 239 (1996).

[36] Trefethen, L.N. *Approximation Theory and Practice.* SIAM, 2013.

[37] Wilkinson, J.H. *The Algebraic Eigenvalue Problem.* Oxford: Oxford University Press, 1965.