STABILITY OF QR-BASED FAST SYSTEM SOLVERS FOR A SUBCLASS OF QUASISEPARABLE RANK ONE MATRICES

FROILÁN M. DOPICO, VADIM OLSHEVSKY, AND PAVEL ZHLOBICH

Abstract. The development of fast algorithms to perform computations with quasiseparable matrices has received a lot of attention in the last decade. Many different algorithms have been presented by several research groups all over the world. Despite this intense activity, to the best of our knowledge, there is no rounding error analysis published for these fast algorithms. In this paper, we present error analyses for two fast solvers of quasiseparable linear systems when they are applied on order one quasiseparable matrices that include the diagonal in the lower triangular rank structure. Both solvers are based on computing first the QR factorization of the coefficient matrix, and their error analyses require novel structured techniques for proving rigorously that only one of the considered algorithms is backward stable, while the other one is not. Two fundamental consequences of this work are: (i) users should employ with caution fast algorithms for quasiseparable matrices since they may be unstable; and (ii) a lot of work has to be done to identify which fast algorithms for quasiseparable matrices are backward stable among the large family available in the literature.

1. Introduction

The study of dense low rank structured matrices and the development of fast algorithms for performing computations with them have attracted much attention in the past decade within Numerical Linear Algebra. Although there are many classical results on low rank structured matrices, the modern interest on these matrices is motivated by their applications in linear system theory and control [5]. It is worth noting that they also appear in many other applications such as oscillatory problems in mechanics, covariance matrices in statistics and the discretization of differential and integral equations [14, Chapter 3]. As an indicator of the intense activity developed in the last years on low rank structured matrices, we mention that the two recent monographs [14, 15] include 500 references in this area, many of them published in the last years. Despite this activity, to the best of our knowledge, there is no rounding error analysis published for the new generation of fast
algorithms for low rank structured matrices. In this context, the main goal of this paper is to start the development of such error analyses by analyzing some algorithms of this type. The most important conclusion that can be obtained from our work is that users should employ these fast algorithms with caution, because their stability is not granted and some of them are not backward stable. We believe that we are opening a vast and difficult field of research, and that a lot of work remains to be done for identifying which fast algorithms for low rank structured matrices are backward stable and/or relevant subclasses of these matrices for which backward stability may be guaranteed.

A first difficulty in analyzing fast algorithms for low rank structured matrices is that these matrices differ in the way they are defined and parameterized, although they always bear the same characteristic property: the presence of large sized submatrices with ranks much smaller than the matrix size. Matrices of this type include quasiseparable, block-quasiseparable, semiseparable, matrices with small Hankel rank, hierarchically semiseparable matrices and others [2, 3, 5, 6, 7, 9, 10, 14, 15]. It would not be inaccurate to say that almost every research group working with low rank structured matrices has its own definition/parametrization. This makes it difficult to compare results obtained for roughly the same matrices but parameterized differently.

In this paper we focus on the rounding error analysis of fast algorithms for solving systems of linear equations whose coefficient matrix is quasiseparable. In particular, we consider algorithms based on computing first the structured QR factorization of this type of matrix. There are several reasons for choosing these algorithms. First, the solution of linear systems is the most basic task in Numerical Linear Algebra and, so, it is the first problem that one should solve in a stable way. For unstructured matrices, Gaussian elimination (GE) with partial pivoting is the standard backward stable linear solver [11]. However, the use of pivoting strategies in GE is forbidden for designing fast linear solvers for quasiseparable matrices, because they destroy the structure and this prevents the design of fast algorithms [14]. On the other hand, there exist fast structured versions of GE without pivoting for quasiseparable matrices [6, 14], but they are are not backward stable. This is not surprising, since it has been known for a long time that fast methods based on GE without pivoting for solving banded systems (banded matrices are a subclass of quasiseparable matrices) are not backward stable in general [11]. By contrast, fast QR-based system solvers for banded matrices are backward stable. Therefore, it is natural to consider fast quasiseparable QR-based system solvers as first candidates to perform rounding error analysis.

In the literature, there are at least two different fast algorithms to solve quasiseparable linear systems via the QR-factorization. For brevity we will refer to them as Algorithms A and B throughout this work. Algorithm A was presented in full generality in [8, Algorithms 6.4 and 7.2], while Algorithm B was essentially introduced in [12, Section 5.3] and it is also explained in [14, Chapter 5]. In fact reference [12] includes two algorithms: a QR-solver and a URV-solver, but both algorithms reduce to a simpler QR-solver for the subclass of matrices that we will consider in this work.

We mention another fast algorithm developed in [2] which is somewhat related to a QR-solver, since it uses orthogonal transformations. However, this algorithm relies on a very different factorization and its error analysis is postponed to a
future work. In [4], the authors develop a QR-based linear solver for a class of rank structured matrices wider than the quasiseparable matrices. This method reduces essentially to Algorithm A for quasiseparable matrices, and, therefore, it is not considered here.

The computational complexity of Algorithm A and Algorithm B is essentially the same and is linear in the size of the matrix. However, they differ in generality: Algorithm B is applicable to order one semiseparable and quasiseparable matrices and Algorithm A covers the more general case of block-quasiseparable matrices of arbitrary order. In addition, each of these algorithms uses a different parametrization of the input quasiseparable matrix. Rounding error analyses have not been developed for either of these algorithms, so it is not known if they are backward stable or not. However, the authors of these algorithms claim that they are stable based on intuitions and numerical experiments (see [8, p. 449] and [12, p. 747]).

Our analysis of Algorithms A and B begins with a simplification: we concentrate on a subclass of order one quasiseparable matrices to which both algorithms are applicable in simplified ways that are amenable for performing rounding error analyses. This subclass is characterized by the property of having the diagonal included in the lower triangular rank structure, i.e., $n \times n$ matrices $A$ in this class satisfy

\[(1.1) \quad \max_{1 \leq i \leq n-1} \text{rank}(A(i:n, 1:i)) = \max_{1 \leq i \leq n-1} \text{rank}(A(1:i, i+1:n)) = 1,\]

where we use MATLAB notation for submatrices. Graphically, such matrices can be represented as follows:

\[
\begin{bmatrix}
\times & \times & \times & \times \\
\times \times & \times & \times \\
\times \times \times & \times \\
\times \times \times \times 
\end{bmatrix},
\]

where every submatrix of a single color has rank at most one. Despite this simplification, our analysis remains valid for matrices that are very relevant in applications, since, in particular, the important class of semiseparable matrices\(^{1}\) satisfies (1.1).

We next discuss the main results of the current paper. We have shown that Algorithm B is not backward stable for matrices satisfying (1.1) by providing a detailed study of this algorithm that shows the potential sources of instabilities, and by constructing counterexamples where these sources of instabilities produce large backward errors. By contrast, Algorithm A turns out to be backward stable for matrices satisfying (1.1). We believe that this is the most relevant result in this work and we have proved it rigorously through a rounding error analysis developed especially for quasiseparable matrices. Two important conclusions can be obtained from these results: (i) there is hope for developing fast algorithms that are simultaneously backward stable for quasiseparable matrices; and (ii) one must be very careful when using currently available fast algorithms for quasiseparable matrices because they may be unstable.

We finally mention that Algorithm A has always produced tiny backward relative errors of order unit roundoff in extensive numerical tests on general order one quasiseparable matrices, i.e., when the diagonal is not included in the rank

\(^{1}\) According to [14] semiseparable matrices are the ones satisfying $\max_i \text{rank}(A(i:n, 1:i)) = \max_i \text{rank}(A(1:i, i+1:n)) = 1$. These matrices are called by other authors Green’s matrices.
structure. However, a rigorous proof of backward stability in this case still remains elusive.

The paper is organized as follows. We start with general ideas on structured QR-decomposition and backward substitution for quasiseparable matrices of type (1.1) in Section 2. In Section 3 we present a specific version of Algorithm A for solving linear systems whose coefficient matrix satisfies (1.1) and its backward stability is proved. We remark that although the error analysis of Algorithm A relies on the quasiseparable structure, the final backward error is not quasiseparable, i.e., we cannot state that the computed solution is the exact solution of a nearby quasiseparable linear system. Section 4 is devoted to Algorithm B: we describe the algorithm, explain the sources of its instability, and give counterexamples that show explicitly that it is not backward stable. Conclusions and future work are discussed in Section 5.

2. General ideas for QR-based system solvers

We assume in this paper that matrices and vectors have real entries and that all matrices are nonsingular. Given an $n \times n$ matrix $A$ with QR factorization $A = QR$ and an $n \times 1$ vector $b$, the unique solution of the linear system $Ax = b$ is the unique solution of the upper triangular system $Rx = Q^Tb$, that can be computed by backward substitution. If $A$ satisfies (1.1), then $Q$ and $R$ have particular structures that allow us to compute in $O(n)$ flops: (i) $Q$ and $R$; (ii) $Q^Tb$; and (iii) the solution of $Rx = Q^Tb$. This is much faster than traditional unstructured methods that cost: $O(n^3)$ for (i), $O(n^2)$ for (ii), and $O(n^2)$ for (iii). Next, the structures of the $Q$ and $R$ factors of matrices satisfying (1.1) are discussed, while specific algorithms will be presented in Sections 3 and 4.

Given a matrix $A$ of type (1.1), every set of subrows in its lower triangular part is linearly dependent, so, we can use one subrow to completely cancel the subrow immediately below until its diagonal entry by using only one Givens rotation. Therefore, to cancel the whole subdiagonal part of $A$ and get $R$ requires only $n - 1$ Givens rotations, $Q$ is a product of $n - 1$ rotations, and $Q$ is an orthogonal lower Hessenberg matrix. It turns out that the rank structure of $R$ differs from the one of the strictly upper triangular part of $A$, because the rank is increased by 1 and the diagonal is included in the structure. These facts are stated in Theorem 2.1, whose proof is omitted since it is closely related to [8, Theorem 7.1] and to results in [14, Chapter 5].

**Theorem 2.1.** Let $A$ be a quasiseparable matrix of order one as in equation (1.1) and let $A = QR$ be its QR factorization, then $Q$ is the product of $n - 1$ Givens rotations and $R$ is quasiseparable of order two, including the diagonal. More precisely, $Q = (W_1W_2\cdots W_{n-1})^T$, $\text{rank}(R(1:i,i:n)) \leq 2$, $1 \leq i \leq n$, where the Givens rotation $W_k$ uses the $k$-th row to cancel out the $k+1$-th row in the strictly lower triangular part of $A$ and has the following structure:

$$
W_k = \begin{bmatrix}
I_{k-1} & c_k & s_k \\
-c_k & I_{n-k-1}
\end{bmatrix}.
$$

(2.1)
In the rest of this section, we explain how the rank structure of $R$ can be used to solve in $O(n)$ operations an upper triangular system $Rx = z$. Note that the backward substitution algorithms that will be used in Sections 3 and 4 are not exactly equal. Therefore, we present here a general algorithm that will be later adapted to the specific representations of $R$ arising in Sections 3 and 4. To this purpose, let us consider now that the diagonal of $R$ is not included in the rank structure, and, in addition, that

$$\max_{1 \leq i \leq n-1} \text{rank}(R(1:i, i+1:n)) = m,$$

with $m < n$ arbitrary, since this fact does not complicate the algorithm at all with respect the case $m = 2$ of interest in this work. The idea of fast quasiseparable backward substitution [14, Section 4.2] is most easily explained via the quasiseparable representation of $R$. It is well known (see [1, 7, 14]) that every upper triangular quasiseparable matrix of order $m$, i.e., satisfying (2.2), admits the following representation:

$$R = \begin{bmatrix}
  d_1 & g_1h_2 & g_1b_2h_3 & \cdots & g_1b_2\ldots b_{n-1}h_n \\
  0 & d_2 & g_2h_3 & \cdots & g_2b_3\ldots b_{n-1}h_n \\
  \vdots & \vdots & \ddots & \ddots & \vdots \\
  0 & 0 & \cdots & 0 & d_n
\end{bmatrix},$$

where the parameters $\{d_k, g_k, b_k, h_k\}$, also called generators, have the following sizes:

- $d_k$: $1 \times 1$
- $g_k$: $1 \times m$
- $b_k$: $m \times m$
- $h_k$: $m \times 1$

Note that, given $R$, the choice of its generators is not unique.

To develop the fast backward substitution algorithm, consider as an example a system of equations with a $5 \times 5$ matrix $R$ given by its generators as in (2.3):

$$\begin{bmatrix}
  d_1 & g_1h_2 & g_1b_2h_3 & g_1b_2b_3h_4 & g_1b_2b_3b_4h_5 \\
  0 & d_2 & g_2h_3 & g_2b_3h_4 & g_2b_3b_4h_5 \\
  0 & 0 & d_3 & g_3h_4 & g_3b_4h_5 \\
  0 & 0 & 0 & d_4 & g_4h_5 \\
  0 & 0 & 0 & 0 & d_5
\end{bmatrix} \begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5
\end{bmatrix} = \begin{bmatrix}
  z_1 \\
  z_2 \\
  z_3 \\
  z_4 \\
  z_5
\end{bmatrix}.$$

Observe that the backward substitution process can be organized as follows,

$$\begin{align*}
x_5 &= (z_5)/d_5, & \tau_4 &= h_5x_5, \\
x_4 &= (z_4 - g_4\tau_4)/d_4, & \tau_3 &= h_4x_4 + b_4\tau_4, \\
x_3 &= (z_3 - g_3\tau_3)/d_3, & \tau_2 &= h_3x_3 + b_3\tau_3, \\
x_2 &= (z_2 - g_2\tau_2)/d_2, & \tau_1 &= h_2x_2 + b_2\tau_2, \\
x_1 &= (z_1 - g_1\tau_1)/d_1,
\end{align*}$$

where an $m \times 1$ temporary vector $\tau$ has been introduced. This is the key to save operations. This idea is generalized directly to $n \times n$ matrices to produce Algorithm 4 which has a cost of $O(m^2n)$ flops — linear in the size of $R$ — due to the product matrix-vector in $\tau$. In fact, the leading term of this cost is $2m^2n$. 


that proves rigorously that \([7, 14]\) of a matrix \([5]\). This specific version is what we call \(3.1\) of the QR-based system solver described first in \([8]\) as an extension of ideas from \([1.1]\). We start this section by presenting a version specific for this type of matrix \((3.1)\), we start this section by presenting a version specific for this type of matrix

\[\begin{align*}
\text{Output: } x_1, x_2, \ldots, x_n
\end{align*}\]

3. Algorithm A

Given a linear system \(Ax = b\), where \(A\) is a nonsingular \(n \times n\) matrix that satisfies \((1.1)\), we start this section by presenting a version specific for this type of matrix of the QR-based system solver described first in \([3]\) as an extension of ideas from \([5]\). This specific version is what we call Algorithm A. Every step will be described in detail, with the purpose of performing in Section \([8.3]\) a rounding error analysis that proves rigorously that Algorithm A is backward stable.

Algorithm A has as inputs the generators of the quasiseparable representation \([7, 14]\) of a matrix \(A\) of type \((1.1)\). This representation is

\[\begin{align*}
\text{(3.1)}
A &=
\begin{bmatrix}
p_1 q_1 & g_1 h_2 & g_1 b_2 h_3 & \cdots & \cdots & g_1 b_2 \ldots b_{n-1} h_n \\
p_2 a_2 q_1 & p_2 q_2 & g_2 h_3 & \cdots & \cdots & g_2 b_3 \ldots b_{n-1} h_n \\
p_3 a_3 a_2 q_1 & p_3 a_2 q_2 & p_3 q_2 & \cdots & \cdots & g_3 b_4 \ldots b_{n-1} h_n \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
p_n a_n \ldots a_2 q_1 & p_n a_n \ldots a_3 q_2 & p_n a_n \ldots a_4 q_3 & \cdots & \cdots & p_n a_n q_{n-1} \quad p_n q_n
\end{bmatrix}
\end{align*}\]

where all the parameters (generators) are scalars.

3.1. Computing the QR factorization. Algorithm 2 below computes the QR factorization of the matrix \(A\) in \((3.1)\) in a fast and concise way. Although Algorithm 2 can be obtained from \([3]\), we will deduce this algorithm, for completeness. To this purpose, recall that Theorem 2.1 states that the factor \(R\) in the QR factorization of \(A\) is quasiseparable of order 2 with the diagonal included in the rank structure. Therefore, \(R\) admits a description via generators of sizes \(1 \times 2, 2 \times 2,\) and \(2 \times 1\), that is, vectors and matrices rather than scalars (recall \((2.2)\) - \((2.3)\)). The first idea to deduce Algorithm 2 is to artificially construct generators with these sizes in the upper triangular part of \(A\) before performing the QR steps. More precisely, the matrix \(A\) from \((3.1)\) is written via the following generator representation:

\[\begin{align*}
\text{(3.2)}
A &=
\begin{bmatrix}
\tilde{g}_0 \tilde{h}_1 & \tilde{g}_0 b_1 \tilde{h}_2 & \tilde{g}_0 b_2 \tilde{h}_3 & \cdots & \cdots & \tilde{g}_0 b_2 \ldots b_{n-1} \tilde{h}_n \\
p_2 a_2 q_1 & \tilde{g}_1 h_2 & \tilde{g}_1 b_2 h_3 & \cdots & \cdots & \tilde{g}_1 b_2 \ldots b_{n-1} \tilde{h}_n \\
p_3 a_3 a_2 q_1 & \tilde{g}_2 h_3 & \tilde{g}_2 h_3 & \cdots & \cdots & \tilde{g}_2 b_3 \ldots b_{n-1} \tilde{h}_n \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
p_n a_n \ldots a_2 q_1 & p_n a_n \ldots a_3 q_2 & p_n a_n \ldots a_4 q_3 & \cdots & \cdots & p_n a_n q_{n-1} \quad p_n q_n
\end{bmatrix}
\end{align*}\]
where
\[
\tilde{g}_k = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad k = 0, 1, \ldots, n - 1,
\]
\[
\tilde{b}_k = \begin{bmatrix} 0 & 0 \\ g_1 & 0 \end{bmatrix}, \quad \tilde{h}_k = \begin{bmatrix} b_k & 0 \\ g_k & 0 \end{bmatrix}, \quad k = 2, \ldots, n - 1,
\]
\[
\bar{h}_1 = \begin{bmatrix} 0 \\ p_1 q_1 \end{bmatrix}, \quad \bar{h}_k = \begin{bmatrix} h_k \\ p_k q_k \end{bmatrix}, \quad k = 2, \ldots, n.
\]

Theorem 2.1 shows that multiplication by \( n - 1 \) Givens rotations is enough to convert \( A \) to the upper triangular form \( R \). These rotations can be computed as follows:
\[
\tilde{p}_n = p_n, \quad \tilde{p}_k = \sqrt{(\tilde{p}_{k+1} a_{k+1})^2 + \tilde{p}_k^2}, \quad k = n - 1, \ldots, 1,
\]
\[
G_k = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}, \quad s_k = \tilde{p}_{k+1} a_{k+1}/\tilde{p}_k, \quad c_k = p_k/\tilde{p}_k,
\]
where the reader may check easily that \( G_k \) corresponds to the Givens transformation \( W_k \) in (2.1) that introduces zeros in the \( k + 1 \)-th row. Observe that the nonsingularity of \( A \) guarantees that \( \tilde{p}_k \neq 0 \), for \( k = n, n - 1, \ldots, 1 \).

Apart from introducing zeros in the strictly lower triangular part, the Givens rotations modify the entries in the upper triangular part of \( A \) and these entries are described by the generators \( \{\tilde{g}_k, \bar{b}_k, \bar{h}_k\} \) according to (3.2). The resulting matrix \( R \) is also described by some generators \( \{\tilde{g}_k, \bar{b}_k, \bar{h}_k\} \) with the same sizes, i.e., \( 1 \times 2, 2 \times 2, \) and \( 2 \times 1 \), respectively. So, the next idea of Algorithm 2 is to find direct computational formulas to relate both sets of parameters

\[
\{\tilde{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^{n} \rightarrow \{\tilde{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^{n},
\]

that are mathematically equivalent to Givens rotations. To deduce these formulas, let us assume for \( k > 1 \) that at every step \( n - 1, n - 2, \ldots, k + 1 \), introducing zeros in rows \( n, n - 1, \ldots, k + 2 \), respectively, the algorithm modified in the upper triangular part only the generators with indices \( n - 1, n - 2, \ldots, k + 1 \). Then at the moment we apply the Givens rotation \( W_k \) to \((W_{k+1} \cdots W_{n-1} A) =: A^{(k)}\) its two subrows \( A^{(k)}(k : k + 1 : k : n) \) are equal to

\[
\begin{bmatrix}
\tilde{g}_k - \tilde{h}_k & \tilde{g}_k b_k + \bar{h}_k g_k & \tilde{g}_k b_k + \bar{h}_k g_k & \cdots & \tilde{g}_k b_k + \bar{h}_k g_k
\\
\tilde{p}_{k+1} a_{k+1} g_k & \tilde{g}_k h_k & \tilde{g}_k h_k & \cdots & \tilde{g}_k h_k
\\
\end{bmatrix}
\begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & h_k & \cdots & \bar{h}_k
\\
\end{bmatrix}.
\]

Hence, the application of \( W_k \) only changes the generators with index \( k \) as follows:

\[
\begin{bmatrix}
c_k & s_k \\
-s_k & c_k
\end{bmatrix}
\begin{bmatrix}
\tilde{g}_k - \tilde{h}_k & \tilde{g}_k - \bar{h}_k \\
\tilde{p}_{k+1} a_{k+1} g_k & \tilde{g}_k
\end{bmatrix}
= \begin{bmatrix}
c_k & s_k \\
-s_k & c_k
\end{bmatrix}
\begin{bmatrix}
p_k q_k & g_k \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
c_k g_k & s_k \\
-s_k g_k & c_k
\end{bmatrix}
= \begin{bmatrix}
\tilde{g}_k - \tilde{h}_k & \tilde{g}_k - \bar{h}_k \\
p_k q_k & 0
\end{bmatrix},
\]

where we introduced the transformed generators

\[
\tilde{g}_k := [s_k g_k, c_k], \quad \tilde{b}_k := [b_k, 0], \quad \tilde{h}_k := [h_k, p_k q_k].
\]
Note that the transformation \( \{ \tilde{g}_k, \tilde{b}_k, \tilde{h}_k \} \rightarrow \{ \tilde{g}_k, \tilde{b}_k, \tilde{h}_k \} \) only changes the rows \( k \) and \( k+1 \) and does not affect any other rows of \( A^{(k)} \), despite the fact that \( \tilde{b}_k \) and \( \tilde{h}_k \) also appear in rows \( 1, \ldots, k-1 \). This follows from the structure of these generators.

Formulas (3.3) are in the heart of Algorithm 2 whose full pseudocode is given below. The computational cost of Algorithm 2 is \( 10n - 9 \) flops.

Algorithm 2. Fast quasiseparable QR factorization, version A.

Input: generators \( \{ p_k \}_{k=1}^n, \{ a_k \}_{k=2}^n, \{ q_k \}_{k=1}^n, \{ g_k \}_{k=1}^{n-1}, \{ b_k \}_{k=2}^{n-1}, \{ h_k \}_{k=2}^n \) of \( A \) in equation (3.1)

\[
\tilde{p}_n = p_n, \quad \tilde{h}_n = \begin{bmatrix} h_n \\ p_n g_n \end{bmatrix}
\]

for \( k = n-1 \) to 2 do

\[
\tilde{p}_k = \sqrt{(\tilde{p}_{k+1} a_{k+1})^2 + p_k^2}, \quad s_k = \frac{\tilde{p}_{k+1} a_{k+1} + \tilde{p}_k}{p_k}, \quad c_k = \frac{p_{k+1}}{p_k}
\]
\[
\tilde{g}_k = [-s_k g_k c_k], \quad \tilde{b}_k = \begin{bmatrix} b_k & 0 \\ c_k g_k & s_k \end{bmatrix}, \quad \tilde{h}_k = \begin{bmatrix} h_k \\ \tilde{p}_k q_k \end{bmatrix}
\]

end for

\[
\tilde{p}_1 = \sqrt{(\tilde{p}_2 a_2)^2 + p_1^2}, \quad s_1 = \frac{\tilde{p}_2 a_2 + \tilde{p}_1}{p_1}, \quad c_1 = \frac{p_2}{p_1}
\]
\[
\tilde{g}_1 = [-s_1 g_1 c_1], \quad \tilde{b}_1 = \begin{bmatrix} 0 & 0 \\ c_1 g_1 & s_1 \end{bmatrix}, \quad \tilde{h}_1 = \begin{bmatrix} 0 \\ \tilde{p}_2 q_1 \end{bmatrix}
\]
\[
\tilde{g}_0 = [0, 1]
\]

Output: \( \{ (c_k, s_k) \}_{k=1}^{n-1} \) cosines and sines of Givens rotations \( W_k \) in (2.1) and generators of the factor \( R \) of \( A \), \( \{ \tilde{g}_k \}_{k=0}^{n-1}, \{ \tilde{b}_k \}_{k=1}^{n-1}, \{ \tilde{h}_k \}_{k=2}^n \), indexed as in (3.4).

Note that the expressions of the generators \( \tilde{b}_1 \) and \( \tilde{h}_1 \) in Algorithm 2 are different than those for \( k > 1 \) in (3.3) because the parametrization (3.1) of \( A \) does not include the scalar parameters \( b_1 \) and \( h_1 \), and \( \tilde{b}_1 \) and \( \tilde{h}_1 \) only appear in the first row of \( R \). We invite the reader to check that the formulas for \( \tilde{b}_1 \) and \( \tilde{h}_1 \) are correct.

Lemma 3.1 states in matrix language that Algorithm 2 produces the right generators of \( R \). This will be useful in the error analysis of Section 3.3. A formal proof of Lemma 3.1 follows by induction on the rows from bottom to top and it is omitted.

Lemma 3.1. Let \( \{ (c_k, s_k) \}_{k=1}^{n-1} \) be the cosines and sines computed by Algorithm 2 corresponding to the Givens rotations \( W_k \) in (2.1) and let \( \{ \tilde{g}_k \}_{k=0}^{n-1}, \{ \tilde{b}_k \}_{k=1}^{n-1}, \{ \tilde{h}_k \}_{k=2}^n \) be the generators computed by Algorithm 2. Assume that all computations are performed in exact arithmetic. Then the matrix

\[
R = \begin{bmatrix}
g_0 \bar{h}_1 & g_0 \bar{b}_1 \bar{h}_2 & g_0 \bar{b}_1 \bar{b}_2 \bar{h}_3 & \cdots & g_0 \bar{b}_1 \cdots \bar{b}_{n-1} \bar{h}_n \\
0 & g_1 \bar{h}_2 & g_1 \bar{b}_2 \bar{h}_3 & \cdots & g_1 \bar{b}_2 \cdots \bar{b}_{n-1} \bar{h}_n \\
0 & 0 & g_2 \bar{h}_3 & \cdots & g_2 \bar{b}_3 \cdots \bar{b}_{n-1} \bar{h}_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & g_{n-2} \bar{h}_{n-1} & g_{n-2} \bar{b}_{n-1} \bar{h}_n \\
0 & 0 & 0 & \cdots & 0 & g_{n-1} \bar{h}_n
\end{bmatrix}
\]

is the \( R \)-factor of the QR factorization of \( A \) in (3.1), that is, \( A = (W_1 \cdots W_{n-1})^T R \).

3.2. Backward substitution and complete Algorithm A. Our next goal is to adapt the general backward substitution Algorithm 1 to upper triangular matrices as those in (3.3). This adaptation yields Algorithm 3 which is valid for any set of
generators \( \{ \tilde{g}_k \}_{k=0}^{n-1}, \{ \tilde{b}_k \}_{k=1}^{n-1}, \{ \tilde{h}_k \}_{k=1}^n \) of sizes 1 \( \times \) 2, 2 \( \times \) 2, and 2 \( \times \) 1, respectively, and not only for generators with the zero-pattern of those computed by Algorithm 2.

To deduce Algorithm 3 simply note that a matrix of type (3.4) can be written as in (2.3) by establishing the correspondences among generators shown in the following table:

<table>
<thead>
<tr>
<th>Generators in (2.3)</th>
<th>Generators in (3.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>{d_1, d_2, \ldots, d_n}</td>
<td>{\tilde{g}_0\tilde{h}_1, \tilde{g}_1\tilde{h}<em>2, \ldots, \tilde{g}</em>{n-1}\tilde{h}_n}</td>
</tr>
<tr>
<td>{h_2, h_3, \ldots, h_n}</td>
<td>{b_1\tilde{h}_2, b_2\tilde{h}<em>3, \ldots, b</em>{n-1}\tilde{h}_n}</td>
</tr>
<tr>
<td>{b_2, b_3, \ldots, b_{n-1}}</td>
<td>{b_1, b_2, \ldots, b_{n-2}}</td>
</tr>
<tr>
<td>{g_1, g_2, \ldots, g_{n-1}}</td>
<td>{\tilde{g}_0, \tilde{g}<em>1, \ldots, \tilde{g}</em>{n-2}}</td>
</tr>
</tbody>
</table>

The use of these correspondences in Algorithm 1 gives Algorithm 3.

**Algorithm 3.** Fast quasiseparable backward substitution to solve \( Rx = z \), version A

**Input:** \( \{z_k\}_{k=1}^n \) and generators \( \{\tilde{g}_k\}_{k=0}^{n-1}, \{\tilde{b}_k\}_{k=1}^{n-1}, \{\tilde{h}_k\}_{k=1}^n \) of \( R \) in (3.4)

for \( k = 1 \) to \( n \)
- \( d_k = \tilde{g}_{k-1}\tilde{h}_k \)
end for

\( x_n = z_n/d_n, \quad \tau = x_n (\tilde{b}_{n-1}\tilde{h}_n) \)

for \( k = n - 1 \) to 2
- \( x_k = (z_k - \tilde{g}_{k-1}\tau)/d_k, \quad \tau = \tilde{b}_{k-1}(x_k\tilde{h}_k + \tau) \)
end for

\( x_1 = (z_1 - \tilde{g}_0\tau)/d_1 \)

**Output:** \( x_1, x_2, \ldots, x_n \)

We write two for-loops in Algorithm 3 to clarify its rounding error analysis in Subsection 3.3 but it is obvious to use only one for-loop. The cost of Algorithm 3 is \( 18n - 16 \) flops for arbitrary generators of order two. For generators computed by Algorithm 2 the cost is \( 16n - 21 \) due to the special zero pattern of these generators.

We have now all the ingredients to state our definitive Algorithm A, whose computational cost is \( 32n - 36 \) flops.

**Algorithm A.** QR-based fast solver of \( Ax = b \) with \( A \) represented as in (3.1)

**Input:** vector \( b \) and generators \( \{p_k\}_{k=1}^n, \{a_k\}_{k=2}^n, \{q_k\}_{k=1}^n, \{g_k\}_{k=1}^{n-1}, \{b_k\}_{k=2}^{n-1}, \{h_k\}_{k=2}^n \) of \( A \) in (3.1)

**Step 1:** Apply Algorithm 2 to compute Givens rotations \( \{W_k\}_{k=1}^n \) and generators of the factor \( R \) of \( A, \{\tilde{g}_k\}_{k=0}^{n-1}, \{\tilde{b}_k\}_{k=1}^{n-1}, \{\tilde{h}_k\}_{k=1}^n \)

**Step 2:** \( z = (W_1 \ldots (W_{n-2}(W_{n-1}b)) \ldots) \)

**Step 3:** Apply Algorithm 3 to solve \( Rx = z \) and compute \( x \)

**Output:** \( x_1, x_2, \ldots, x_n \)
3.3. Rounding error analysis of Algorithm A. In this section, we perform first error analyses of Algorithms 2 and 3. Then, we prove in Theorem 3.5 that Algorithm A is backward stable. This is the most important result in this work.

We make common assumptions about the model of floating point arithmetic [11, Ch. 2]. Given two floating point numbers \( x \) and \( y \), then
\[
\text{fl}(x \text{ op } y) = \frac{x \text{ op } y}{1 + \alpha}, \quad |\alpha| \leq u, \quad \text{op} = +, -, *, /.
\]
(3.5)

where \( u \) is the unit roundoff, which is of order \( 10^{-16} \) in IEEE double precision arithmetic. We will also use the following result and standard notation for accumulating rounding errors: if \( |\delta_i| \leq u \) and \( \rho_i = \pm 1 \) for \( i = 1, \ldots, k \), and \( ku < 1 \), then
\[
\prod_{i=1}^{k} (1 + \delta_i)^{\rho_i} = 1 + \Theta_k, \quad |\Theta_k| \leq \frac{ku}{1 - ku} =: \gamma_k
\]
for any positive integer \( k \). These symbols will be used according to the rules presented in [11, Ch. 3]. We define \( \Theta_0 = \gamma_0 = 0 \). As usual, “hats” are employed to denote quantities computed in floating point arithmetic.

Next, we introduce additional notation. The Euclidean norm in \( \mathbb{R}^n \) is denoted by \( \| \cdot \|_2 \), as well as the matrix spectral norm, i.e., the maximum singular value of a matrix. The matrix Frobenius norm will be denoted by \( \| \cdot \|_F \). For matrices \( B \) and \( C \) (and vectors) inequalities \( B \leq C \) mean \( b_{ij} \leq c_{ij} \) for all \( (i,j) \) entries. Absolute values are also understood in componentwise sense: \( |B| \) is the matrix with entries \( |b_{ij}| \).

Theorem 3.2 below deals with backward errors in Algorithm 2 for computing the QR factorization of \( A \) in (3.1). First of all, we remark that Theorem 3.2 is not a standard backward error result, so we provide some comments to make easier its understanding. Recall that the key idea of backward error analysis is to prove that the outputs computed by a certain algorithm in floating point arithmetic are exact outputs for slightly perturbed inputs [11]. In this sense, one should naturally try in our problem to attach backward errors to the generators of \( A \) in (3.1). We have not been able to do this, and we doubt it is possible. Instead, in part (a) of Theorem 3.2 we attach tiny relative backward errors to each column of the matrix \( A \). This destroys the structure of \( A \), i.e., \( A + E \) is close to \( A \) but does not have any particular structure. Moreover, this forces us to state the result in terms of the upper triangular matrix \( R \) exactly constructed from the generators computed by Algorithm 2. This is a matrix that is never constructed in Algorithm A and is only used for the theoretical purpose of proving that Algorithm 2 is “unstructured” backward stable. However, we have found that this is not enough to prove backward stability of the whole Algorithm A. Therefore, in part (b) of Theorem 3.2 it is proved that small columnwise backward errors in \( A \) are preserved if \( R \) is perturbed via tiny componentwise variations, that may be different for each column of \( R \), of the computed generators. Part (b) is the key to match the backward errors in Algorithm 2 with those of Algorithm 2.

**Theorem 3.2.** Let \( \{\tilde{g}_k\}_{k=1}^{n-1}, \{\tilde{b}_k\}_{k=1}^{n-1}, \) and \( \{\tilde{h}_k\}_{k=1}^{n} \) be the generators computed in floating point arithmetic by Algorithm 2 for the upper triangular QR factor of \( A \) in (3.1). Then the following two statements hold.
(a) Let $R_{\hat{g}}$ be the upper triangular matrix exactly constructed from $\{\hat{g}_k\}_{k=0}^{n-1}$, $\{\hat{b}_k\}_{k=1}^{n-1}$, $\{\hat{h}_k\}_{k=1}^{n-1}$ according to (3.4). Then there exists an exact orthogonal $n \times n$ matrix $Q$ such that

$$A + E = Q R_{\hat{g}}, \quad \text{where } \|E(:,j)\|_2 \leq \gamma C_1 n \|A(:,j)\|_2, \quad j = 1, \ldots, n,$$

and $C_1$ is a small integer constant. The matrix $Q$ is given explicitly as $Q = (W_1 W_2 \cdots W_{n-1})^T$, where $W_k$ is an exact Givens rotation with the structure given in (2.1) and based on the exact sine and cosine

$$s'_k = \hat{p}_{k+1} a_{k+1}/\sqrt{(\hat{p}_{k+1} a_{k+1})^2 + p_k^2}, \quad c'_k = p_k/\sqrt{(\hat{p}_{k+1} a_{k+1})^2 + p_k^2},$$

constructed with the computed parameter $\hat{p}_{k+1}$.

(b) Let $R_{g+\delta g}$ be any $n \times n$ upper triangular matrix defined as follows

$$(R_{g+\delta g})_{ij} = \begin{cases} \hat{g}_{j-1}^{(i)} \hat{h}_{j}^{(i)} & \text{if } i = j, \\ \hat{g}_{j-1}^{(i)} \hat{b}_{j}^{(i)} \cdots \hat{b}_{j-1}^{(i)} \hat{h}_{j}^{(i)} & \text{if } i < j, \\ 0 & \text{if } i > j, \end{cases}$$

where, for a certain nonnegative integer $p$,

$$|\hat{g}_{j}^{(i)} - \hat{g}_{i}| \leq \gamma_p |\hat{g}_{i}| \quad \text{for } \ 1 \leq j \leq n, \ 0 \leq i \leq j - 1,$$

$$|\hat{b}_{j}^{(i)} - \hat{b}_{i}| \leq \gamma_p |\hat{b}_{i}| \quad \text{for } \ 1 \leq j \leq n, \ 1 \leq i \leq j - 1,$$

$$|\hat{h}_{j}^{(i)} - \hat{h}_{i}| \leq \gamma_p |\hat{h}_{i}| \quad \text{for } \ 1 \leq j \leq n.$$

Then the matrix $Q$ in part (a) satisfies

$$A + F = Q R_{g+\delta g}, \quad \text{where } \|F(:,j)\|_2 \leq \gamma C_1 n + C_{2p^n} \|A(:,j)\|_2, \quad j = 1, \ldots, n,$$

and $C_2$ is a small integer constant.

Proof. Part (a) follows from part (b) by taking $p = 0$, i.e., $\gamma_0 = 0$. Therefore, we only prove part (b).

First, the errors in sines and cosines are established. At every step $i = n-1, \ldots, 1$ of Algorithm 2, sines and cosines computed in floating point arithmetic satisfy

$$\tilde{s}_i = fl\left(\hat{p}_{i+1} a_{i+1}/\sqrt{(\hat{p}_{i+1} a_{i+1})^2 + p_i^2}\right) = s'_i(1 + \Theta_6),$$

where $s'_i = \hat{p}_{i+1} a_{i+1}/\sqrt{(\hat{p}_{i+1} a_{i+1})^2 + p_i^2}$,

$$\tilde{c}_i = fl\left(p_i/\sqrt{(\hat{p}_{i+1} a_{i+1})^2 + p_i^2}\right) = c'_i(1 + \Theta_5),$$

where $c'_i = p_i/\sqrt{(\hat{p}_{i+1} a_{i+1})^2 + p_i^2}$.

So, $s'_i$ and $c'_i$ are exact corresponding sines and cosines.

Similarly to the error analysis of classical Householder or Givens algorithms for the QR decomposition [11 Ch. 19], the rest of the proof is columnwise and focuses in a fixed column $A(:,j)$ of the matrix $A$ in (3.1). The index $j$ remains constant in the proof. We split the proof into two steps.

Step 1. Here, we analyze the errors that are produced in $A(:,j)$ by the Givens rotations that introduce zeros in rows $n, n-1, \ldots, j+1$. These correspond to
\[ (c_{n-1}, s_{n-1}), (c_{n-2}, s_{n-2}), \ldots, (c_j, s_j). \] For \( i = n - 1, \ldots, j \) the computed quantities \( \hat{p}_i \) satisfy:

\[
\hat{p}_i = f(I) \left( \sqrt{\left( \hat{p}_{i+1} a_{i+1} \right)^2 + p_i^2} \right) = (1 + \Theta_4) \sqrt{\left( \hat{p}_{i+1} a_{i+1} \right)^2 + p_i^2} = \left[ c'_i(1 + \Theta_4) \quad s'_i(1 + \Theta_4) \right] \left[ \hat{p}_i \atop \hat{p}_{i+1} a_{i+1} \right].
\]

Hence,

\[
\tag{3.7}
\begin{bmatrix}
\hat{p}_i \\
0
\end{bmatrix} = \begin{bmatrix} c'_i(1 + \Theta_4) & s'_i(1 + \Theta_4) \\
-s'_i & c'_i
\end{bmatrix} \begin{bmatrix} p_i \\
\hat{p}_{i+1} a_{i+1}
\end{bmatrix} = (G_i + \Delta G_i) \begin{bmatrix} p_i \\
\hat{p}_{i+1} a_{i+1}
\end{bmatrix},
\]

where

\[
\tag{3.8}
G_i := \begin{bmatrix} c'_i & s'_i \\
-s'_i & c'_i
\end{bmatrix}, \quad \text{and} \quad \Delta G_i := \begin{bmatrix} c'_i \Theta_4 & s'_i \Theta_4 \\
0 & 0
\end{bmatrix}.
\]

In order to simplify the notation, here and thereafter let us agree to denote by \( G_i \) a \( 2 \times 2 \) Givens rotation, as well as any Givens rotation of other appropriate size based on \((c'_i, s'_i)\) and that when it is applied to subvectors \( A(k : l, j) \) of the \( j \)-th column only modifies those entries corresponding to rows \( i \) and \( i + 1 \) of the input matrix \( A \). The same agreement is valid for the error matrices \( \Delta G_i \), with the additional property that all entries of \( \Delta G_i \) are zero except those corresponding to the submatrix \((c'_i, s'_i); -s'_i, c'_i)\) in \( G_i \). These agreements simplify the typesetting considerably.

Multiplying (3.7) by \( a_i \ldots a_j q_j \) in exact arithmetic we get:

\[
\begin{bmatrix} \hat{p}_i a_i \ldots a_{j+1} q_j \\
0
\end{bmatrix} = (G_i + \Delta G_i) \begin{bmatrix} p_i a_i \ldots a_{j+1} q_j \\
\hat{p}_{i+1} a_{i+1} a_i \ldots a_{j+1} q_j
\end{bmatrix},
\]

which, by induction, leads to

\[
\tag{3.9}
\begin{bmatrix} \hat{p}_j q_j \\
0 \\
\vdots \\
0
\end{bmatrix} = \prod_{i=j}^{n-1} (G_i + \Delta G_i) \begin{bmatrix} p_j q_j \\
p_{j+1} a_{j+1} q_j \\
\vdots \\
p_{n+1} a_{n+1} \ldots a_{j+1} q_j
\end{bmatrix} = \prod_{i=j}^{n-1} (G_i + \Delta G_i) A(j : n, j).
\]

Note that the matrices \( G_i \) are the same for each column \( j \).

**Step 2.** Here, we analyze the errors that are produced in \( A(:, j) \) by the Givens rotations that introduce zeros in rows \( j, j-1, \ldots, 2 \). Let us look at the computed generators in Algorithm 2 that contribute to column \( R_{ij}(:, j) \):

\[
\begin{align*}
\hat{g}_i &= \left[ -s_i g_i (1 + \Theta_1) \quad c_i \right] = \left[ -s'_i g_i (1 + \Theta_7) \quad c'_i (1 + \Theta_5) \right], \quad 0 \leq i \leq j - 1, \\
\hat{b}_i &= \begin{bmatrix} b_i \\
0
\end{bmatrix} = \begin{bmatrix} b_i \\
0
\end{bmatrix}, \quad 1 \leq i \leq j - 1, \\
\hat{h}_j &= \begin{bmatrix} h_j \\
\hat{p}_j q_j (1 + \Theta_1')
\end{bmatrix},
\end{align*}
\]

where, for simplicity, we define \( g_0 = s_0 = s'_0 = b_1 = h_1 = 0 \) and \( c_0 = c'_0 = 1 \) to take into account the special structure of generators \( \hat{g}_0, \hat{b}_1 \) and \( \hat{h}_1 \). Note also that the \( \Theta_m \) quantities that appear in the equations above depend on the index \( i \) of the generators. We drop this dependence by simplicity. According to the definition of generators \( \hat{g}_i^{(j)}, \hat{b}_i^{(j)} \) and \( \hat{h}_i^{(j)} \) in the statement of part (b) of Theorem 2, they...
are obtained from \( \tilde{g}_i, \tilde{b}_i, \) and \( \tilde{h}_j \) by introducing tiny componentwise perturbations bounded by \( \gamma_p \). Therefore, we can write

\[
\tilde{g}_{i}^{(j)} = \begin{bmatrix} -s_i' g_i (1 + \Theta_{p+7}) & c_i'(1 + \Theta_{p+5}) \end{bmatrix}, \quad 0 \leq i \leq j - 1,
\]

\[
\tilde{b}_{i}^{(j)} = \begin{bmatrix} b_i (1 + \Theta_p) & 0 \\ c_i' g_i (1 + \Theta_{p+6}) & s_i' (1 + \Theta_{p+6}) \end{bmatrix}, \quad 1 \leq i \leq j - 1,
\]

\[
\tilde{h}_j^{(j)} = \begin{bmatrix} h_j (1 + \Theta'_p) \\ \tilde{p}_j q_j (1 + \Theta_{p+1}) \end{bmatrix}.
\]

Our task now is to relate, in exact arithmetic, the nonzero entries of the \( j \)-th column of \( R_{\tilde{g}+\tilde{b}} \) with the entries of the matrix \( A \) in (3.1) whose generators are the inputs of Algorithm 2 Recall that

\[
\bar{r}_{ij} := (R_{\tilde{g}+\tilde{b}})_{ij} = \tilde{g}_{i-1}^{(j)} \tilde{h}_{i}^{(j)} \ldots \tilde{b}_{j-1}^{(j)} \tilde{h}_{j}^{(j)}, \quad \text{if } i < j,
\]

and \( \bar{r}_{jj} := (R_{\tilde{g}+\tilde{b}})_{jj} = \tilde{g}_{j-1}^{(j)} \tilde{h}_{j}^{(j)} \). Note that all the entries \( \bar{r}_{ij} \), for \( i = j, \ldots, 1 \), can be computed by multiplying generators from the end to the beginning as follows:

\[
\tau_j = \tilde{h}_j^{(j)},
\]

\[
\bar{r}_{ij} = \tilde{g}_{i-1}^{(j)} \tau_i, \quad \tau_i = \tilde{b}_{i-1}^{(j)} \tau_i, \quad i = j, \ldots, 2,
\]

\[
\bar{r}_{1j} = \tilde{g}_{0}^{(j)} \tau_1,
\]

here \( \tau_i = [\tau_{i,1} \tau_{i,2}]^T \) is an auxiliary column vector of length two.

Let us note that each step of (3.11), for \( i = j, \ldots, 2 \), can be written in matrix form (by using (3.10)) as

\[
\begin{bmatrix} \tau_{i-1,2} \\ \bar{r}_{ij} \end{bmatrix} = \begin{bmatrix} c_{i-1}' (1 + \Theta_{p+6}) & s_{i-1}' (1 + \Theta_{p+6}) \\ -s_{i-1}' (1 + \Theta_{p+7}) & c_{i-1}' (1 + \Theta_{p+5}) \end{bmatrix} \begin{bmatrix} g_{i-1} \tau_{i,1} \\ \tau_{i,2} \end{bmatrix},
\]

\[
\tau_{i-1,1} = b_{i-1} \tau_{i,1} (1 + \Theta_p).
\]

For \( i = j \), we know that \( \tau_j = \tilde{h}_j^{(j)} \) and, so,

\[
\begin{bmatrix} \tau_{j-1,2} \\ \bar{r}_{jj} \end{bmatrix} = \begin{bmatrix} c_{j-1}' (1 + \Theta_{2p+6}) & s_{j-1}' (1 + \Theta_{2p+6}) \\ -s_{j-1}' (1 + \Theta_{2p+7}) & c_{j-1}' (1 + \Theta_{2p+5}) \end{bmatrix} \begin{bmatrix} g_{j-1} h_j (1 + \Theta'_p) \\ \tilde{p}_j q_j (1 + \Theta_{p+1}) \end{bmatrix},
\]

\[
\tau_{j-1,1} = b_{j-1} h_j (1 + \Theta_{2p}),
\]

or, attaching errors to the Givens rotation,

\[
\begin{bmatrix} \tau_{j-1,2} \\ \bar{r}_{jj} \end{bmatrix} = \begin{bmatrix} c_{j-1}' (1 + \Theta_{2p+6}) & s_{j-1}' (1 + \Theta_{2p+7}) \\ -s_{j-1}' (1 + \Theta_{2p+7}) & c_{j-1}' (1 + \Theta_{2p+5}) \end{bmatrix} \begin{bmatrix} g_{j-1} h_j \\ \tilde{p}_j q_j \end{bmatrix},
\]

\[
\tau_{j-1,1} = b_{j-1} h_j (1 + \Theta_{2p}).
\]

So, \( \tau_{j,1} = h_j (1 + \Theta'_p), \) \( \tau_{j-1,1} = b_{j-1} h_j (1 + \Theta_{2p}), \) and by induction

\[
\tau_{i,1} = b_i b_{i+1} \ldots b_{j-1} h_j (1 + \Theta_{(j-i+1)p}), \quad \text{for } i = j - 1, \ldots, 1.
\]

Using this to change \( \tau_{i,1} \) in equation (3.12) we get

\[
\begin{bmatrix} \tau_{i-1,2} \\ \bar{r}_{ij} \end{bmatrix} = \begin{bmatrix} c_{i-1}' (1 + \Theta_{(j-i+2)p+6}) & s_{i-1}' (1 + \Theta_{(j-i+2)p+6}) \\ -s_{i-1}' (1 + \Theta_{(j-i+2)p+7}) & c_{i-1}' (1 + \Theta_{p+5}) \end{bmatrix} \begin{bmatrix} g_{i-1} b_i \ldots b_{j-1} h_j \\ \tau_{i,2} \end{bmatrix}.
\]
Recall the simplified notation introduced in (3.8) and the paragraph just below it and stick equations (3.13) together for \( i = 2, \ldots, j \):

\[
\begin{bmatrix}
\tau_{1,2} \\
\tau_{2j} \\
\vdots \\
\vdots \\
\tau_{jj}
\end{bmatrix} = \prod_{i=1}^{j-1} (G_i + \Delta G_i) = \prod_{i=1}^{j-1} (G_i + \Delta G_i) \begin{bmatrix}
g_1 b_2 \ldots b_j h_j \\
g_2 b_3 \ldots h_j \\
\vdots \\
g_{j-1} h_j \\
p_j q_j
\end{bmatrix} = \prod_{i=1}^{j-1} (G_i + \Delta G_i) \begin{bmatrix}
A(1; j-1,j) 
\end{bmatrix}.
\]

We still need to relate \( \tau_{1,2} \) to \( \tau_{1j} \). But this is trivial, since (3.11) and \( \tilde{g}_0 = [0 \ 1] \) imply \( \tau_{1j} = (1 + \Theta_p)\tau_{1,2} \). Therefore, one can replace \( \tau_{1,2} \) in (3.14) by \( \tau_{1j} \) at the cost of attaching the extra error \( 1 + \Theta_p \) to the first row of \( G_1 + \Delta G_1 \), and so modifying the first row of \( \Delta G_1 \).

Equations (3.9) and (3.14) (with \( \tau_{1,2} \) replaced by \( \tau_{1j} \) as explained above) lead to

\[
R_{\tilde{g} + \delta g}(; j) = \prod_{i=1}^{n-1} \left( W_i + \Delta W_i \right) A(;j),
\]

where \( W_1, \ldots, W_{n-1} \) are the Givens rotations defined in part (a) of the statement of Theorem 3.2. Equations (3.13) and (3.14) define \( \Delta W_i \) and imply that

\[
\begin{align*}
\| \Delta W_i \|_F &\leq \sqrt{2}\gamma_{(j+1)p+6} \| W_i \|_2, \\
\| \Delta W_i \|_F &\leq \sqrt{2}\gamma_{(j-i+1)p+7} \| W_i \|_2, \\
\| \Delta W_i \|_F &\leq \gamma_4 \| W_i \|_2, 
\end{align*}
\]

\( i = 2, \ldots, j-1 \)

\( i = j, \ldots, n-1 \)

Therefore, using \( Q^T = \prod_{i=1}^{n-1} W_i \) and (11 Lemma 3.7), we get

\[
R_{\tilde{g} + \delta g}(; j) = \left( \prod_{i=1}^{n-1} W_i + \prod_{i=1}^{n-1} (W_i + \Delta W_i) - \prod_{i=1}^{n-1} W_i \right) A(;j) = Q^T (A(;j) + F(;j)),
\]

with

\[
\| F(;j) \|_2 \leq \left\| \prod_{i=1}^{n-1} (W_i + \Delta W_i) - \prod_{i=1}^{n-1} W_i \right\|_F \| A(;j) \|_2 \leq \gamma_{c_{1n} + c_{2pn}^2} \| A(;j) \|_2,
\]

which proves the result.

\[ \square \]

Observe that parts (a) and (b) of Theorem 3.2 imply that \( R_{\tilde{g} + \delta g}(; j) - R_{\tilde{g}}(;j) = Q^T (F(;j) - E(;j)) \) and, therefore,

\[
\| R_{\tilde{g} + \delta g}(;j) - R_{\tilde{g}}(;j) \|_2 \leq \gamma_{2c_{1n} + c_{2pn}^2} \| A(;j) \|_2,
\]

for \( j = 1, \ldots, n \). This means that tiny relative componentwise perturbations of the generators of \( R_{\tilde{g}}(;j) \) produce tiny relative variations in the whole \( j \)-th column, i.e., the generators determine well the columns of \( R_{\tilde{g}} \), although perhaps not the entries.

Let us make an interesting remark. In the assertion of part (a) of Theorem 3.2 we assumed that the matrix \( R_{\tilde{g}} \) is constructed in exact arithmetic from the computed generators. This is sufficient for our purposes, as we never construct it and only store its generators. On the other hand, if one wants to compute the entries of \( R_{\tilde{g}} \)

\[ 2 \text{The reader should observe that in step 2 of the proof, that is, for } i = 1, \ldots, j-1, \text{ the backward errors in Givens rotations are } \Delta G_i = [c'_i \Theta_{(j-i+1)p+6} ; s'_i \Theta_{p+6} ; -s'_i \Theta_{(j-i+1)p+7} ; c'_i \Theta_{p+5}]. \]
explicitly in floating point arithmetic, we can also assure backward stability as it is shown in Corollary 3.3 below. This is a consequence of part (b) in Theorem 3.2.

**Corollary 3.3.** Let \( \{\hat{g}_k\}_{k=0}^{n-1}, \{\hat{b}_k\}_{k=1}^{n-1}, \{\hat{h}_k\}_{k=1}^n \) be the generators computed in floating point arithmetic by Algorithm 2 for the upper triangular QR factor of \( A \) in (3.1). Let \( Q \) and \( R_{\hat{g}} \) be the matrices in the statement of Theorem 3.2. Consider the following iteration to compute, for \( j = 1, \ldots, n \), the entries \( \hat{r}_{jj}, \hat{r}_{j-1,j}, \ldots, \hat{r}_{1j} \) of \( R_{\hat{g}} \):

\[
\begin{align*}
\tau_j &= \hat{h}_j, \\
\hat{r}_{ij} &= \hat{g}_{i-1}\tau_i, & \tau_{i-1} &= \hat{b}_{i-1}\tau_i, & i &= j, \ldots, 2, \\
\hat{r}_{1j} &= \hat{g}_0\tau_1,
\end{align*}
\]

where \( \tau_i \) is an auxiliary column vector of length two. Let the upper triangular matrix \( \hat{R}_{\hat{g}} \) be the output of (3.16) in floating point arithmetic. Then

\[
Q\hat{R}_{\hat{g}} = A + \Delta A, \quad \text{where} \quad \|\Delta A(:,j)\|_2 \leq \gamma_{Cn^2}\|A(:,j)\|_2, \quad j = 1, \ldots, n,
\]

where \( C \) is a small integer constant.

**Proof.** We only sketch the proof. Standard error analysis of matrix-vector multiplication \[11\] Sect. 3.5] applied to the iteration (3.16) for computing the \( j \)-th column of \( \hat{R}_{\hat{g}} \) introduce componentwise relative backward errors, i.e., perturbations, bounded by \( \gamma_2 \) in generators \( \hat{g}_k \) and \( \hat{b}_k \). Although these perturbations will be different for each column \( j \), part (b) in Theorem 3.2 can be used with \( p = 2 \), which gives the result. \( \square \)

In fact, it is possible to prove several addenda to Corollary 3.3. For instance, if \( \hat{Q} \) is the orthogonal factor of the QR factorization of \( A \) computed in floating point arithmetic by multiplying explicitly the Givens rotations, then \( \hat{Q}\hat{R}_{\hat{g}} = A + \Delta A \), with \( \Delta A \) bounded as in Corollary 3.3 with slightly different constants. Moreover, it can be proved that \( \|\hat{R}_{\hat{g}}(:,j) - R_{\hat{g}}(:,j)\|_2 \leq \gamma_{C'n^2}\|R_{\hat{g}}(:,j)\|_2 \), for all \( j \). This means that \( R_{\hat{g}} \) is computed explicitly with tiny relative forward errors.

Next, we analyze in Theorem 3.4 backward errors in Algorithm 3. Recall that Algorithm 3 is valid for any set of input generators of sizes \( 1 \times 2, 2 \times 2, \) and \( 2 \times 1 \). The same holds for the error analysis. Observe that the analysis attaches backward errors to the input generators, and that for the generators \( \hat{g}_k \) the errors are different for different columns.

**Theorem 3.4.** Let \( R \) be an \( n \times n \) upper triangular matrix given as in (3.1) by its generators \( \{\hat{g}_k\}_{k=0}^{n-1}, \{\hat{b}_k\}_{k=1}^{n-1}, \{\hat{h}_k\}_{k=1}^n \) of sizes \( 1 \times 2, 2 \times 2, \) and \( 2 \times 1 \), respectively. Suppose the system \( Rx = z \) is solved in floating point arithmetic using Algorithm 3. Then the computed solution \( \hat{x} \) is the exact solution of

\[
(3.17) \quad R_g + \delta_g \hat{x} = z,
\]
where the matrix $R_{g+\delta g}$ is defined as
\[
R_{g+\delta g} = \begin{bmatrix}
\tilde{g}_0\tilde{h}_1 & \tilde{g}_0\tilde{b}_1\tilde{h}_2 & \tilde{g}_0\tilde{b}_2\tilde{h}_3 & \cdots & \tilde{g}_0\tilde{b}_1\tilde{b}_{n-1}\tilde{h}_n \\
0 & \tilde{g}_1\tilde{h}_2 & \tilde{g}_1\tilde{b}_2\tilde{h}_3 & \cdots & \tilde{g}_1\tilde{b}_2\tilde{b}_{n-1}\tilde{h}_n \\
0 & 0 & \tilde{g}_2\tilde{h}_3 & \cdots & \tilde{g}_2\tilde{b}_3\tilde{b}_{n-1}\tilde{h}_n \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & \tilde{g}_{n-2}\tilde{h}_{n-1} \tilde{g}_{n-2}\tilde{b}_{n-1}\tilde{h}_n \\
0 & 0 & 0 & \cdots & 0 & \tilde{g}_{n-1}\tilde{h}_n
\end{bmatrix}
\]
in terms of certain generators $\{\tilde{g}_k\}_{k=0}^{n-2} \subset \mathbb{R}^{1 \times 2}$, $\{\tilde{g}_k\}_{k=0}^{n-1} \subset \mathbb{R}^{1 \times 2}$, $\{\tilde{b}_k\}_{k=1}^{n-1} \subset \mathbb{R}^{2 \times 2}$, and $\{\tilde{h}_k\}_{k=1}^{n} \subset \mathbb{R}^{2 \times 1}$ that satisfy
\[
\begin{align*}
|\tilde{g}_k - \bar{g}_k| &\leq \gamma_2 |\bar{g}_k|, \quad \text{for } k = 0, \ldots, n-2, \\
|\tilde{g}_k - \bar{g}_k| &\leq \gamma_5 |\bar{g}_k|, \quad \text{for } k = 0, \ldots, n-1, \\
|\tilde{b}_k - \bar{b}_k| &\leq \gamma_3 |\bar{b}_k|, \quad \text{for } k = 1, \ldots, n-1, \\
|\tilde{h}_k - \bar{h}_k| &\leq u |\bar{h}_k|, \quad \text{for } k = 1, \ldots, n.
\end{align*}
\]

Proof. It is easy to check that if the first for-loop is removed from Algorithm 3 and $\{d_1, d_2, \ldots, d_n\}$ are given as (arbitrary) input parameters, then Algorithm 3 would compute the solution of
\[
(3.20) \quad Tx = z,
\]
where $t_{ij} = r_{ij}$ for $i \neq j$, and $t_{ii} = d_i$ for $i = 1, \ldots, n$.

Let $\tilde{d}_k = fl(\bar{g}_{k-1}\tilde{h}_k)$, for $k = 1, \ldots, n$, be the numbers computed in floating point arithmetic in the first for-loop of Algorithm 3. Then, standard error analysis [11 Ch. 3] gives that the computed solution $\tilde{x}$ satisfies
\[
\begin{align*}
\tilde{x}_n &= \frac{z_n}{d_n(1 + \varepsilon_n)}, \quad |\varepsilon_n| \leq u, \\
|\tilde{\tau}_{n-1} - \bar{\tau}_{n-1}| &= \frac{\tilde{x}_n(\bar{b}_{n-1} + \Delta\bar{b}_{n-1})\tilde{h}_n}{d_n(1 + \varepsilon_n)(1 + \varepsilon'_n)}, \quad |\varepsilon_n|, |\varepsilon'_n| \leq u, \quad |\Delta\bar{b}_{n-1}| \leq \gamma_3 |\bar{b}_{n-1}|, \\
\tilde{x}_k &= \frac{\tilde{z}_k - (\tilde{g}_{k-1} + \Delta\tilde{g}_{k-1})\tilde{\tau}_{k}}{\tilde{d}_k(1 + \varepsilon_k)(1 + \varepsilon'_k)}, \quad |\varepsilon_k|, |\varepsilon'_k| \leq u, \quad |\bar{g}_{k-1}| \leq \gamma_2 |\tilde{g}_{k-1}|, \\
|\tilde{\tau}_{k-1} - \bar{\tau}_{k-1}| &= \frac{\tilde{\tau}_{k-1}(\tilde{b}_{k-1} + \Delta\tilde{b}_{k-1})}{\tilde{x}_k(\tilde{h}_k + \Delta\tilde{h}_k + \tilde{\tau}_{k})}, \quad |\Delta\bar{b}_{k-1}| \leq \gamma_3 |\bar{b}_{k-1}|, |\Delta\bar{h}_k| \leq u |\bar{h}_k|,
\end{align*}
\]
for
\[
\begin{align*}
\tilde{x}_1 &= \frac{z_1 - (\bar{g}_0 + \Delta\bar{g}_0)\tilde{\tau}_1}{\tilde{d}_1(1 + \varepsilon_1)(1 + \varepsilon'_1)}, \quad |\varepsilon_1|, |\varepsilon'_1| \leq u, \quad |\Delta\bar{g}_0| \leq \gamma_2 |\tilde{g}_0|.
\end{align*}
\]
Comparing this recurrence with Algorithm 3 recalling (3.20), and defining $\varepsilon'_n := 0$, we get that $\tilde{x}$ is the exact solution of
\[
R_{g+\delta g} \tilde{x} = z,
\]
where the entries of $R_{g+\delta g}$ are given by
\[
(R_{g+\delta g})_{ij} = \begin{cases}
\tilde{d}_i(1 + \varepsilon_i)(1 + \varepsilon'_i) & \text{if } i = j, \\
(\tilde{g}_{i-1} + \Delta\tilde{g}_{i-1})(\tilde{b}_i + \Delta\tilde{b}_i) \cdots (\tilde{b}_{j-1} + \Delta\tilde{b}_{j-1})(\tilde{h}_{j} + \Delta\tilde{h}_j) & \text{if } i < j, \\
0 & \text{if } i > j.
\end{cases}
\]

3We introduce indexes in the computed versions of the auxiliary vectors $\tau$ to clarify the analysis.
To finish the proof of Theorem 3.4 we first define \( \bar{g}_i := \hat{g}_i + \Delta \hat{g}_i \), for \( i = 0, \ldots, n-2 \), \( \bar{b}_i = b_i + \Delta h_i \), for \( i = 1, \ldots, n-1 \), and \( \bar{h}_i = h_i + \Delta h_i \), for \( i = 1, \ldots, n \) (we take \( \Delta h_n = \Delta h_1 = 0 \)). Second, note that, for \( k = 1, \ldots, n \), if \( \bar{g}_{k-1} = [\bar{g}_{k-1}^1 \quad \bar{g}_{k-1}^2] \), then

\[
\bar{d}_k = \begin{bmatrix} (1 + \Theta_2) & \bar{g}_{k-1}^2 (1 + \Theta_2') \end{bmatrix} \bar{h}_k.
\]

In addition, \( \tilde{h}_k = \begin{bmatrix} (1 + \delta_1) & 0 \\ 0 & (1 + \delta_2) \end{bmatrix} \bar{h}_k \), with \( |\delta_1| \leq u, |\delta_2| \leq u \). From (3.21), we get \( \tilde{d}_k = \begin{bmatrix} \bar{g}_{k-1}^1 (1 + \Theta_2) & \bar{g}_{k-1}^2 (1 + \Theta_2') \end{bmatrix} \bar{h}_k \), and

\[
(R_{g+\delta g})_{kk} = \tilde{d}_k (1 + \varepsilon_k) (1 + \varepsilon'_k) = \begin{bmatrix} \bar{g}_{k-1}^1 (1 + \Theta_3) & \bar{g}_{k-1}^2 (1 + \Theta_3') \end{bmatrix} \bar{h}_k \equiv \bar{g}_{k-1} h_k.
\]

This completes the proof. \( \square \)

Using part (b) of Theorem 3.2 and Theorem 3.4 together we can finally prove the main result in this paper, that is, columnwise backward stability of Algorithm A.

**Theorem 3.5 (Algorithm A is backward stable).** Let \( A \) be an \( n \times n \) quasiseparable matrix given by its generators \( \{p_k\}_{k=1}^n, \{a_k\}_{k=2}^n, \{g_k\}_{k=1}^{n-1}, \{b_k\}_{k=2}^n, \{h_k\}_{k=2}^n \) as in (3.1), and let \( b \in \mathbb{R}^n \). Suppose the system \( Ax = b \) is solved in floating point arithmetic using Algorithm A. Then the computed solution \( \hat{x} \) is the exact solution of

\[
(A + \Delta A)\hat{x} = b + \Delta b,
\]

where

\[
\|\Delta A(:,j)\|_2 \leq \gamma_{Knz} \|A(:,j)\|_2, \quad j = 1, \ldots, n, \quad \|\Delta b\|_2 \leq \gamma_{K2n} \|b\|_2,
\]

and \( K_1 \) and \( K_2 \) are small integer constants.

**Proof.** Let \( \hat{z} \) be the vector computed in step 2 of Algorithm A. Then Theorem 3.4 can be applied to step 3 of Algorithm A, which implies that \( \hat{z} \) is the exact solution of a system of linear equations

\[
R_{\hat{g}+\delta g} \hat{x} = \hat{z},
\]

where the matrix \( R_{\hat{g}+\delta g} \) is defined as in (3.19), with the difference that in (3.19) appear the generators \( \{\tilde{g}_k\}_{k=0}^{n-1}, \{\tilde{b}_k\}_{k=1}^{n-1}, \{\tilde{h}_k\}_{k=1}^n \) computed in step 1 of Algorithm A instead of \( \{\hat{g}_k\}_{k=1}^n, \{\hat{b}_k\}_{k=2}^n, \{\hat{h}_k\}_{k=1}^n \). Observe that part (b) of Theorem 3.2 can be applied with \( p = 5 \) to \( R_{\hat{g}+\delta g} \) in (3.22) and we get

\[
Q^T (A + \Delta A) \hat{x} = \hat{z}, \quad \text{where} \quad \|\Delta A(:,j)\|_2 \leq \gamma_{Kn} \|A(:,j)\|_2, \quad j = 1, \ldots, n,
\]

where \( Q \) is the exact orthogonal matrix defined in Theorem 3.2. It only remains to analyze the errors in the computation of \( \hat{z} \). Recall that, according to (3.9), Givens rotations we compute in Algorithm 2 satisfy

\[
\hat{G}_i = \begin{bmatrix} \hat{c}_i & \hat{s}_i \\ -\hat{s}_i & \hat{c}_i \end{bmatrix} = \begin{bmatrix} c'_i (1 + \Theta_5) & s'_i (1 + \Theta_6) \\ -s'_i (1 + \Theta_6) & c'_i (1 + \Theta_5) \end{bmatrix},
\]

where \( s'_i \) and \( c'_i \) are exact sines and cosines corresponding to the Givens rotation \( \mathcal{W}_i \) defined in part (a) of Theorem 3.2. Standard error analysis of multiplication by a sequence of Givens rotations (see Lemmas 19.7 and 19.9 in [11]) yields

\[
\hat{z} = (\mathcal{W}_1 \mathcal{W}_2 \cdots \mathcal{W}_{n-1}) (b + \Delta b) = Q^T (b + \Delta b), \quad \|\Delta b\|_2 \leq \gamma_{K2n} \|b\|_2.
\]
Combining this equation with (3.23) gives \( Q^T(A + \Delta A)\bar{x} = Q^T(b + \Delta b) \), and so \((A + \Delta A)\bar{x} = b + \Delta b\). This proves the result. \( \square \)

4. Algorithm B

This QR-based solver for linear systems was first presented in [12, Section 5.3] for semiseparable-plus-diagonal matrices and later in [14, pp. 194–199] for general quasiseparable matrices of order one. In this section, we will apply the later version of the algorithm to the subclass of order one quasiseparable matrices defined by equation (1.1). This allows us to obtain the simple linear solver that we call Algorithm B. Numerical tests and counterexamples presented in Subsection 4.2 show that even in this simple case the algorithm in [14] fails to compute backward stable solutions and we identify in Subsection 4.3 which are the intrinsic sources of instability of Algorithm B.

The algorithm described in [14] used the so-called Givens-vector representation [13]. For the matrices defined by (1.1), this parametrization is [14, p. 195]:

\[
A = A_L + A_U = \begin{bmatrix}
    c_1 v_1 & 0 & 0 & \cdots & 0 & 0 \\
    c_2 s_1 v_1 & c_2 v_2 & 0 & \cdots & \vdots & \vdots \\
    c_3 s_2 s_1 v_1 & c_3 s_2 v_2 & c_3 v_3 & \cdots & \vdots & \vdots \\
    \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
    c_{n-1} s_{n-2} \cdots s_1 v_1 & \cdots & c_{n-1} v_{n-1} & 0 & \vdots & \vdots \\
    s_{n-1} s_{n-2} \cdots s_1 v_1 & \cdots & s_{n-1} v_{n-1} & v_n & \vdots & \vdots \\
\end{bmatrix}
\]

(4.1)

Here all parameters are scalars and \((c_i, s_i)\) and \((r_i, t_i)\) are pairs of cosines and sines of some angles. The absolute values of \(v_i\) are equal to the corresponding norms of the columns of \(A_L\), i.e., \(\|A_L(:,i)\|_2 = |v_i|\). Similarly, the absolute values of the parameters \(e_i\) are norms of rows of \(A_U\). So, the overall set of parameters describing matrix \(A\) is

\[
\begin{align*}
    G &= \begin{bmatrix}
        c_1 & c_2 & \cdots & c_{n-1} \\
        s_1 & s_2 & \cdots & s_{n-1}
    \end{bmatrix}, \\
    v &= [v_1 \ v_2 \ \cdots \ v_n]^T, \\
    H &= \begin{bmatrix}
        r_1 & r_2 & \cdots & r_{n-2} \\
        t_1 & t_2 & \cdots & t_{n-2}
    \end{bmatrix}, \\
    e &= [e_1 \ e_2 \ \cdots \ e_{n-1}]^T.
\end{align*}
\]

Observe that the Givens-vector representation (4.1) assumes, in fact, that we already know the factor \(Q\) in the QR factorization of \(A\). The matrix \(Q\) is given by its Schur representation established in Theorem 2.1 \(Q^T = W_1 W_2 \cdots W_{n-1}\), where \(W_i\) is the \(n \times n\) Givens transformation in (2.1) based on the pair \((c_i, s_i)\).

The only open question to get the QR factorization of \(A\) is what is the representation of the factor \(R\). The authors of [14] answer this question as follows. Applying the rotations \(W_i\) to \(A\) they show that \(R\) can be written as the sum of two upper
triangular semiseparable matrices of order one. We will repeat their construction using a $4 \times 4$ example for simplicity. We assume as in [13] that there are no divisions by zero.

The original $4 \times 4$ matrix $A$ is

$$
A = A_L + A_U = \begin{bmatrix}
  c_1 v_1 & 0 & 0 & 0 \\
  c_2 s_1 v_1 & c_2 v_2 & 0 & 0 \\
  c_3 s_2 s_1 v_1 & c_3 s_2 v_2 & c_3 v_3 & 0 \\
  s_3 s_2 s_1 v_1 & s_3 s_2 v_2 & s_3 v_3 & v_4
\end{bmatrix}
+ \begin{bmatrix}
  0 & r_1 e_1 & r_2 t_1 e_1 & t_2 t_1 e_1 \\
  0 & 0 & r_2 e_2 & t_2 e_2 \\
  0 & 0 & 0 & e_3 \\
  0 & 0 & 0 & 0
\end{bmatrix}.
$$

Now we apply Givens transformations to eliminate rows in the strictly lower triangular part of $A$ one by one starting from the bottom. The first transformation is $W_3$:

$$
(4.3) \quad W_3 A = W_3 A_L + W_3 A_U
= \begin{bmatrix}
  c_1 v_1 & 0 & 0 & 0 \\
  c_2 s_1 v_1 & c_2 v_2 & 0 & 0 \\
  s_2 s_1 v_1 & s_2 v_2 & \overline{v_3} & s_3 \overline{v_4} \\
  0 & 0 & 0 & c_3 \overline{v_4}
\end{bmatrix}
+ \begin{bmatrix}
  0 & r_1 e_1 & r_2 t_1 e_1 & t_2 t_1 e_1 \\
  0 & 0 & r_2 e_2 & t_2 e_2 \\
  0 & 0 & 0 & c_3 e_3 \\
  0 & 0 & 0 & e_4
\end{bmatrix},
$$

where we introduced new parameters: $\overline{v_4} = v_4$ and $\overline{e_4} = -s_3 e_3$, whose absolute values are the norms of the last column and the last row in $W_3 A_L$ and $W_3 A_U$, respectively. Our ultimate goal is to have $R$ as the sum of two upper triangular semiseparable matrices of order one each at the end of applying all Givens transformations. After the first step, the upper triangular part of matrix $W_3 A_L$ satisfies this “wish” but matrix $W_3 A_U$ does not, because the following $2 \times 2$ block in its upper triangular part

$$
\begin{bmatrix}
  r_2 e_2 & t_2 e_2 \\
  0 & c_3 e_3
\end{bmatrix}
$$

has rank two in general. Fortunately, we can modify the zero entry in this $2 \times 2$ block to change the rank to one. Let us note that this zero entry in matrix $W_3 A_U$ is located at exactly the same position as the entry $v_3$ in matrix $W_3 A_L$ (see equation (4.3)) and $v_3$ does not appear anywhere else in the later matrix. Hence, we are free to add an adequate number $r_2 x_3$ to $0$ in the entry $(3, 3)$ of $W_3 A_U$ and subtract it from $v_3$ in $W_3 A_L$. The choice of $x_3$ that makes the last two columns of the upper triangular part of $W_3 A_U$ semiseparable of order one is $x_3 = c_3 e_3 / t_2$.

If we also introduce a new parameter $\overline{v_3} = v_3 - r_2 x_3$, then equation (4.3) becomes:

$$
(4.4) \quad W_3 A = A_L^{(3)} + A_U^{(3)}
= \begin{bmatrix}
  c_1 v_1 & 0 & 0 & 0 \\
  c_2 s_1 v_1 & c_2 v_2 & 0 & 0 \\
  s_2 s_1 v_1 & s_2 v_2 & \overline{v_3} & s_3 \overline{v_4} \\
  0 & 0 & 0 & c_3 \overline{v_4}
\end{bmatrix}
+ \begin{bmatrix}
  0 & r_1 e_1 & r_2 t_1 e_1 & t_2 t_1 e_1 \\
  0 & 0 & r_2 e_2 & t_2 e_2 \\
  0 & 0 & 0 & r_2 x_3 \\
  0 & 0 & 0 & \overline{e_4}
\end{bmatrix}.
$$
Applying the next Givens transformation we get
\[
W_2W_3A = W_2A_L^{(3)} + W_2A_U^{(3)}
\]
\[
= \begin{bmatrix}
c_1 v_1 & 0 & 0 & 0 \\
s_1 v_1 & \tilde{v}_2 & s_2 \tilde{v}_3 & s_2 s_3 \tilde{v}_4 \\
0 & 0 & c_2 \tilde{v}_3 & c_2 s_3 \tilde{v}_4 \\
0 & 0 & 0 & c_3 \tilde{v}_4
\end{bmatrix}
+ \begin{bmatrix}
0 & r_1 e_1 & r_2 t_1 e_1 & t_2 t_1 e_1 \\
0 & 0 & r_2 (c_2 e_2 + s_2 x_3) & t_2 (c_2 e_2 + s_2 x_3) \\
0 & 0 & r_2 (c_2 x_3 - s_2 e_2) & t_2 (c_2 x_3 - s_2 e_2) \\
0 & 0 & 0 & \tilde{e}_4
\end{bmatrix},
\]
where we again need to change the highlighted zero entry to some number \(r_1 x_2\) to produce semiseparability in the upper triangular part of \(W_2A_U^{(3)}\):

\[
(4.5) \quad W_2W_3A = A_L^{(2)} + A_U^{(2)}
\]
\[
= \begin{bmatrix}
c_1 v_1 & 0 & 0 & 0 \\
s_1 v_1 & \tilde{v}_2 & s_2 \tilde{v}_3 & s_2 s_3 \tilde{v}_4 \\
0 & 0 & c_2 \tilde{v}_3 & c_2 s_3 \tilde{v}_4 \\
0 & 0 & 0 & c_3 \tilde{v}_4
\end{bmatrix}
+ \begin{bmatrix}
0 & r_1 e_1 & r_2 t_1 e_1 & t_2 t_1 e_1 \\
0 & 0 & r_1 x_2 & t_1 x_2 \\
0 & 0 & r_2 \tilde{e}_3 & t_2 \tilde{e}_3 \\
0 & 0 & 0 & \tilde{e}_4
\end{bmatrix},
\]
with
\[
\tilde{e}_3 = c_2 x_3 - s_2 e_2, \quad x_2 = \frac{1}{t_1}(c_2 e_2 + s_2 x_3), \quad \tilde{v}_2 = v_2 - r_1 x_2.
\]
At this point only one Givens transformation is left, its application leads to \(4.6\), this time without the need of adding and subtracting any quantity in the (1,1)-entry.

\[
(4.6) \quad W_1W_2W_3A = W_1A_L^{(2)} + W_1A_U^{(2)}
\]
\[
= \begin{bmatrix}
v_1 & s_1 \tilde{v}_2 & s_1 s_2 \tilde{v}_3 & s_1 s_2 s_3 \tilde{v}_4 \\
0 & c_1 \tilde{v}_2 & c_1 s_2 \tilde{v}_3 & c_1 s_2 s_3 \tilde{v}_4 \\
0 & 0 & c_2 \tilde{v}_3 & c_2 s_3 \tilde{v}_4 \\
0 & 0 & 0 & c_3 \tilde{v}_4
\end{bmatrix}
+ \begin{bmatrix}
0 & r_1 \tilde{e}_1 & r_2 t_1 \tilde{e}_1 & t_2 t_1 \tilde{e}_1 \\
0 & 0 & r_1 \tilde{e}_2 & t_1 \tilde{e}_2 \\
0 & 0 & r_2 \tilde{e}_3 & t_2 \tilde{e}_3 \\
0 & 0 & 0 & \tilde{e}_4
\end{bmatrix},
\]
where \(\tilde{e}_1 = c_1 e_1 + s_1 x_2\) and \(\tilde{e}_2 = c_1 x_2 - s_1 e_1\).

All together we have the QR decomposition of matrix \(A\) expressed as:
\[A = QR,\] where \(Q = W_3^TW_2^TW_1^T\) and \(R = R_L + R_U\),
where \(R_L := W_1A_L^{(2)}\), \(R_U := W_1A_U^{(2)}\) are given in \(4.3\) and both of them are triangular order one semiseparable matrices, whose rank structure captures the diagonal.

The procedure described above trivially generalizes to \(n \times n\) matrices \(A\) represented in Givens-vector form as in \(4.1\). This procedure is presented as Algorithm \(4\), The cost of Algorithm \(4\) is \(9n - 12\) flops, and its output is the set of parameters that allows us to reconstruct the \(R\) factor in the QR factorization of \(A\), with \(R\) given, according to \(4.6\), as a sum of two triangular matrices \(R_L\) and \(R_U\) of the form

\[
(4.7) \quad R_L = \begin{bmatrix}
\tilde{v}_1 & s_1 \tilde{v}_2 & s_1 s_2 \tilde{v}_3 & \cdots & s_1 s_2 \cdots s_{n-2} \tilde{v}_{n-1} & s_1 s_2 \cdots s_{n-1} \tilde{v}_n \\
0 & c_1 \tilde{v}_2 & c_1 s_2 \tilde{v}_3 & \cdots & c_1 s_2 \cdots s_{n-2} \tilde{v}_{n-1} & c_1 s_2 \cdots s_{n-1} \tilde{v}_n \\
0 & 0 & c_2 \tilde{v}_3 & \cdots & c_2 s_3 \cdots s_{n-2} \tilde{v}_{n-1} & c_2 s_3 \cdots s_{n-1} \tilde{v}_n \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & \cdots & c_{n-2} \tilde{v}_{n-1} & c_{n-2} s_{n-1} \tilde{v}_n \\
0 & 0 & \cdots & \cdots & 0 & c_{n-1} \tilde{v}_n
\end{bmatrix}
\]
and

\[(4.8) \quad R_U = \begin{bmatrix} r_1 \tilde{e}_1 & r_2 t_1 \tilde{e}_1 & \cdots & r_{n-2} t_{n-3} \cdots t_1 \tilde{e}_1 & t_{n-2} \cdots t_1 \tilde{e}_1 \\ 0 & r_2 \tilde{e}_2 & r_2 t_1 \tilde{e}_2 & \cdots & r_{n-2} t_{n-3} \cdots t_1 \tilde{e}_2 & t_{n-2} \cdots t_1 \tilde{e}_2 \\ 0 & 0 & r_2 \tilde{e}_3 & \cdots & r_{n-2} t_{n-3} \cdots t_2 \tilde{e}_3 & t_{n-2} \cdots t_2 \tilde{e}_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & r_{n-2} \tilde{e}_{n-1} & t_{n-2} \tilde{e}_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & \tilde{e}_n \end{bmatrix} \]

Algorithm 4. Fast quasiseparable QR factorization, version B.

**Input:** 
Givens-vector parameters \( \{c_k\}_{k=1}^{n-1}, \{s_k\}_{k=1}^{n-1}, \{v_k\}_{k=1}^n, \{r_k\}_{k=1}^{n-2}, \{t_k\}_{k=1}^{n-2} \)
and \( \{e_k\}_{k=1}^n \) of \( A \) in (4.1).

\[x_n = 0, \quad \tilde{v}_n = v_n\]

**for** \( k = n-1 \) **to** 2 **do**

\[
\begin{bmatrix} x_k \\ e_{k+1} \end{bmatrix} = \begin{bmatrix} t_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} e_k \\ x_{k+1} \end{bmatrix}
\]

\[
\tilde{v}_k = v_k - r_{k-1} x_k
\]

**end for**

\[
\begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \end{bmatrix} = \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} e_1 \\ x_2 \end{bmatrix}
\]

**end for**

\[\tilde{v}_1 = v_1\]

**Output:** parameters \( \{\tilde{v}_k\}_{k=1}^n, \{\tilde{e}_k\}_{k=1}^n \) of \( R_L \) and \( R_U \) in (4.7)–(4.8).

In order to solve fast the system of equations \( Ax = b \), we still need to find a fast solver of the upper triangular linear system \( (R_L + R_U)x = Q^T b \), in terms of the generators of \( R_L \) and \( R_U \) appearing in (4.7)–(4.8). To this purpose note that the matrix \( R = R_L + R_U \) is quasiseparable of order 2 and, hence, admits the representation \( \{2.3\} \). Therefore, we simply compute in Algorithm 5 the generators of \( R \) in \( \{2.3\} \) and then apply Algorithm 1. The cost of Algorithm 5 is \( 5n - 7 \) flops.

Algorithm 5. Computing generators of \( R = R_L + R_U \) from generators of \( R_L \) and \( R_U \).

**Input:** generators \( \{c_k\}_{k=1}^{n-1}, \{s_k\}_{k=1}^{n-1}, \{v_k\}_{k=1}^n, \{r_k\}_{k=1}^{n-2}, \{t_k\}_{k=1}^{n-2} \) of \( R_L \) in (4.7) and generators \( \{\tilde{v}_k\}_{k=1}^n, \{\tilde{e}_k\}_{k=1}^n \) of \( R_U \) in (4.8).

\[d_1 = \tilde{v}_1, \quad g_1 = \begin{bmatrix} s_1 \\ \tilde{e}_1 \end{bmatrix}\]

**for** \( k = 2 \) **to** \( n-1 \) **do**

\[
d_k = c_{k-1} \tilde{v}_k + r_{k-1} \tilde{e}_k, \quad g_k = \begin{bmatrix} c_{k-1} s_k \\ t_{k-1} \tilde{e}_k \end{bmatrix} \\
b_k = \begin{bmatrix} s_k & 0 \\ 0 & t_{k-1} \end{bmatrix}, \quad h_k = \begin{bmatrix} \tilde{v}_k \\ r_{k-1} \end{bmatrix}
\]

**end for**

\[
d_n = c_{n-1} \tilde{v}_n + \tilde{e}_n, \quad h_n = \begin{bmatrix} \tilde{v}_n \\ 1 \end{bmatrix}
\]

**Output:** generators \( \{d_k\}_{k=1}^n, \{g_k\}_{k=1}^n, \{b_k\}_{k=2}^n, \{h_k\}_{k=2}^n \) of \( R \) as in \( \{2.3\} \).

We have now all the ingredients to state the definitive Algorithm B, whose cost is \( 31n - 39 \) flops. To deduce this cost, we have taken into account that the
generators $b_k$ of $R$ are diagonal matrices, which reduces in this case the cost of Algorithm 1. Observe that the cost of Algorithm B is a little bit less than the cost of Algorithm A, which is $32n - 36$ flops. This is in part related to the fact that the Givens-vector representation has as inputs the pairs cosine-sine corresponding to the Givens rotations of the factor $Q$, while Algorithm A needs to compute these quantities.

Algorithm B. QR-based fast solver of $Ax = b$ with $A$ represented as in (4.1).

Input: vector $b$ and Givens-vector parameters $\{c_k\}_{k=1}^{n-1}$, $\{s_k\}_{k=1}^{n-1}$, $\{v_k\}_{k=1}^n$, $\{r_k\}_{k=1}^{n-2}$, $\{t_k\}_{k=1}^{n-2}$, and $\{e_k\}_{k=1}^{n-1}$ of $A$ in (4.1).

Step 1: Apply Algorithm 4 to compute parameters $\{\tilde{v}_k\}_{k=1}^{n}$, $\{\tilde{e}_k\}_{k=1}^{n}$ of $R_L$ and $R_U$ in (4.7)–(4.8).

Step 2: $z = (W_1... (W_{n-2}(W_{n-1}b))...)$, where $W_i$ is a Givens rotation based on the input cosine-sine pair $(c_i, s_i)$.

Step 3: Apply Algorithm 5 to compute the generators of $R = R_L + R_U$ represented as in (2.3).

Step 4: Apply Algorithm 1 to solve $Rz = x$ and compute $x$.

Output: $x_1, x_2, \ldots, x_n$.

4.1. Potential sources of instability of Algorithm B. In this section we discuss in detail an important source of backward instability of Algorithm B. This instability appears in step 1, i.e., in Algorithm 4 and is an intrinsic effect of representing the factor $R$ as a summation of two matrices that may have very different norms. The analysis is informal, but we will illustrate that this source of instability may have a dramatic effect in the numerical tests presented in Subsection 4.2.

Observe that Algorithm 4 simply computes norms (up to signs) $\{\tilde{v}_1, \ldots, \tilde{v}_n\}$ of columns of $R_L$ and norms (up to signs) $\{\tilde{e}_1, \ldots, \tilde{e}_n\}$ of rows of $R_U$, since the rest of generators of $R_L$ and $R_U$ are some of the input parameters. The QR factorization of $A$ in exact arithmetic in terms of $R_L$ and $R_U$ is

$$A = Q(R_L + R_U).$$

In floating point arithmetic the factor $Q$ is given exactly by the input parameters, as long as it is stored in the factorized form $Q = W_{n-1}^T \cdots W_2^T W_1^T$, but some errors are necessarily committed to obtain the computed norms $\{\tilde{v}_1, \ldots, \tilde{v}_n\}$ and $\{\tilde{e}_1, \ldots, \tilde{e}_n\}$. Therefore, if $\tilde{R}_L$ and $\tilde{R}_U$ are the matrices obtained in exact arithmetic when one replaces $\tilde{v}_i$ by $\hat{v}_i$ and $\tilde{e}_i$ by $\hat{e}_i$, for $i = 1, \ldots, n$, in (4.7) and (4.8), respectively, then the parameters computed by Algorithm 4 correspond to the QR factorization of

$$\hat{B} = Q(\tilde{R}_L + \tilde{R}_U).$$

This is the exact QR factorization of a certain matrix $\hat{B}$. Therefore, the backward error for Algorithm 4 is

$$A - \hat{B} = Q(R_L - \tilde{R}_L + R_U - \tilde{R}_U).$$

The unstructured Givens algorithm to compute the QR factorization gives normwise relative backward errors of order unit roundoff, $\mathbf{u}$, in each column 11. Theorem
so let us bound the Euclidean norms of the columns of the backward error in (4.10):

\[ \frac{\|A(:,i) - \hat{B}(;i)\|_2}{\|A(:,i)\|_2} \leq \frac{\|R_L(:,i) - \hat{R}_L(:,i)\|_2}{\|R_L(:,i)\|_2} \frac{\|R_L(:,i)\|_2}{\|A(:,i)\|_2} \]

(4.11)

\[ + \frac{\|R_U(:,i) - \hat{R}_U(:,i)\|_2}{\|R_U(:,i)\|_2} \frac{\|R_U(:,i)\|_2}{\|A(:,i)\|_2}, \quad i = 1, \ldots, n. \]

The bound (4.11) shows that, even in the best possible ideal scenario in which Algorithm 4 would produce error,

\[ \frac{\|R_L(:,i) - \hat{R}_L(:,i)\|_2}{\|R_L(:,i)\|_2} \approx u \quad \text{and} \quad \frac{\|R_U(:,i) - \hat{R}_U(:,i)\|_2}{\|R_U(:,i)\|_2} \approx u, \]

the normwise relative backward error for columns in Algorithm 4 may be huge if

(4.12) \[ \max \left\{ \frac{\|R_L(:,i)\|_2}{\|A(:,i)\|_2}, \frac{\|R_U(:,i)\|_2}{\|A(:,i)\|_2} \right\} \geq 1, \quad \text{for some } i. \]

In addition, if (4.12) holds for some index \( j \) such that \( \|A(:,j)\|_2 \approx \|A\|_2 \), then the whole backward error \( \|A - \hat{B}\|_2/\|A\|_2 \) may be huge. We have seen in the numerical tests of Subsection 4.2 that this often happens.

We want to stress that if the condition (4.12) is satisfied for a column \( i \), then equation (4.9) is not a reliable representation of \( A(:,i) \) under perturbations, because \( A(:,i) \) is given as a summation of two vectors \( QR_L(:,i) \) and \( QR_U(:,i) \) that have much larger norms than \( A(:,i) \). Therefore tiny relative perturbations of \( QR_L(:,i) \) and/or \( QR_U(:,i) \) may produce enormous relative variations in the norm of \( A(:,i) \). This is an example of the well-known phenomenon of severe cancellation.

In the rest of this subsection we explain why Algorithm 4 may produce outputs that satisfy (4.12). Note that Algorithm 4 has a recursive form. To proceed with the analysis, we need to obtain the direct mapping of input parameters \( \{c_k\}_{k=1}^{n-1}, \{s_k\}_{k=1}^{n-1}, \{v_k\}_{k=1}^{n}, \{r_k\}_{k=1}^{n-2}, \{t_k\}_{k=1}^{n-2}, \{e_k\}_{k=1}^{n-1} \) to output parameters \( \{\tilde{v}_k\}_{k=1}^{n}, \{\tilde{e}_k\}_{k=1}^{n} \). Let us introduce the block-diagonal transformation matrices

(4.13) \[ \Theta_k = \text{diag} \left( I_{k-1}, \begin{bmatrix} 1 & 0 \\ t_{k-1} & 1 \end{bmatrix} \right) \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}, I_{n-k-1} \),

where for \( k = 1 \), we define \( t_0 = 1 \). Then, by the virtue of Algorithm 4

\[
\begin{bmatrix}
e_1 \\
\vdots \\
e_{n-2} \\
x_{n-1} \\
\tilde{e}_n
\end{bmatrix} = \Theta_{n-1} \begin{bmatrix}
e_1 \\
\vdots \\
e_{n-2} \\
e_{n-1} \\
0
\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}
\tilde{e}_1 \\
\vdots \\
\tilde{e}_{n-1} \\
\tilde{e}_n
\end{bmatrix} = \Theta_1 \cdots \Theta_{n-1} \begin{bmatrix}
e_1 \\
\vdots \\
e_{n-1} \\
0
\end{bmatrix}.
\]

Hence, if we define vectors \( \tilde{e} = [\tilde{e}_1 \cdots \tilde{e}_n]^T \) and \( e = [e_1 \cdots e_{n-1} \ 0]^T \), then

\[ \|\tilde{e}\|_2 \leq \prod_{k=1}^{n-1} \|\Theta_k\|_2 \cdot \|e\|_2 = \frac{\|e\|_2}{|t_1 t_2 \ldots t_{n-2}|}. \]

Recall that the numbers \( t_k \) are sines of certain angles, so \( |1/t_k| > 1 \) and it may happen that \( \|\tilde{e}\|_2 \gg \|e\|_2 \). To simplify the discussion, let all \( t_k \) be of magnitude \( t \),
then in the worst case \( \| \tilde{e} \|_2 \) would grow exponentially with \( n \): \( \| \tilde{e} \|_2 \approx \left( \frac{1}{2} \right)^{n-2} \| e \|_2 \).

This is also \( \| \tilde{e} \|_2 \approx 2^{n-2} \| e \|_2 \), for the moderate value \( t = 1/2 \). In this case, observe that

\[
\| R_U(:,i) \|_2 \geq |e_i| = \frac{\sqrt{3}}{2} |\tilde{e}_i| \quad \text{for} \quad i = 2, \ldots, n - 1.
\]

Therefore, for \( |\tilde{e}_k| = \max_i |\tilde{e}_i|, \) assuming \( 2 \leq k \leq n - 1 \), we have for the worst case of growth of \( \| \tilde{e} \|_2 \)

\[
\begin{align*}
\| R_U(:,k) \|_2 & \geq \frac{\sqrt{3}}{2\sqrt{n}} \| \tilde{e} \|_2 \\
& \approx \frac{\sqrt{3}}{\sqrt{n}} 2^{n-3} \| e \|_2 \\
& = \frac{\sqrt{3}}{\sqrt{n}} 2^{n-3} \| A_U \|_F \\
& \approx \frac{\sqrt{3}}{\sqrt{n}} 2^{n-3} \| A \|_F \\
& \geq \frac{\sqrt{3}}{\sqrt{n}} 2^{n-3} \| A(:,k) \|_2,
\end{align*}
\]

where we have assumed that

\[
\| A_L \|_F \approx \| A_U \|_F.
\]

Observe that this worst case scenario leads to an exponential growth with the size of the matrix of \( \| R_U(:,k) \|_2 / \| A(:,k) \|_2 \).

4.2. Numerical tests showing that Algorithm B is unstable. This section includes examples where Algorithm B computes solutions \( \hat{x} \) with large relative backward errors for \( Ax = b \). As usual, we employ the relative residual \( \| A\hat{x} - b \|_2 / \| A \|_2 \| \hat{x} \|_2 \)
to compute the minimum normwise relative backward error, since it is known [11, Chapter 7] that

\[
\| A\hat{x} - b \|_2 / \| A \|_2 \| \hat{x} \|_2 = \min \{ \varepsilon : (A + \Delta A)\hat{x} = b, \quad \| \Delta A \|_2 \leq \varepsilon \| A \|_2 \}.
\]

Traditional unstructured solvers, like Gaussian elimination with partial pivoting or QR, give \( \| A\hat{x} - b \|_2 / \| A \|_2 \| \hat{x} \|_2 \approx u \approx 10^{-16} \), in double precision, for any condition number of \( A \). We have performed our numerical tests in MATLAB, i.e., in IEEE double precision.

We start by constructing a very small size counterexample by choosing one of the generators \( t_k \) tiny as follows:

\[
G = \begin{bmatrix}
\cos(\pi/6) & \cos(\pi/3) & \cos(10^{-6}) \\
\sin(\pi/6) & \sin(\pi/3) & \sin(10^{-6})
\end{bmatrix}, \quad
v = [1 \ 1 \ 1]^T,
\]

\[
H = \begin{bmatrix}
\cos(\pi/4) & \cos(10^{-6}) \\
\sin(\pi/4) & \sin(10^{-6})
\end{bmatrix}, \quad
e = [1 \ 1]^T.
\]

According to (4.1), these generators give the matrix:

\[
A = \begin{bmatrix}
0.8660 & 0.7071 & 0.7071 & 0.7071 \cdot 10^{-6} \\
0.2500 & 0.5000 & 0.9999 & 1.0000 \cdot 10^{-6} \\
0.4330 & 0.8660 & 0.9999 & 1.0000 \\
0.4330 \cdot 10^{-6} & 0.8660 \cdot 10^{-6} & 1.0000 \cdot 10^{-6} & 1.0000
\end{bmatrix},
\]

with \( \text{cond}(A) = 16.8185 \). The right-hand side \( b \) of the system is chosen to be \( b = [1 \ 1 \ 1]^T \). Then Algorithm B produces the relative residual

\[
\eta = \frac{\| A\hat{x} - b \|_2}{\| A \|_2 \| \hat{x} \|_2} = 1.2644 \cdot 10^{-11},
\]

which is much larger than \( u \). To understand this fact, note that for this example, \( \| A \|_2 = 2.3011, \| R_L \|_2 \approx \| R_U \|_2 = 1.2788 \cdot 10^6 \), where \( R_L \) and \( R_U \) were defined in
and $M = 5.5575 \cdot 10^5$, with

$$M := \max_i \max_k \left\{ \frac{\|R_L(:,i)\|_2}{\|A(:,i)\|_2}, \frac{\|R_U(:,i)\|_2}{\|A(:,i)\|_2} \right\},$$

i.e., the maximum of the magnitudes in (4.12). Observe that $\eta \approx M \cdot u$. This is precisely the source of instability discussed in Subsection 4.1.

The next test illustrates that Algorithm B may produce large relative residuals for moderate values of the sines $t_k$. We construct, for $n = 10, 20, \ldots, 90$, matrices $n \times n$ with parameters $t_k = 1/2$, for $k = 1, \ldots, n-2$, $s_k = \sqrt{2}/2$, for $k = 1, \ldots, n-1$, $v_k = 1$, for $k = 1, \ldots, n$, and $e_k = 1$, for $k = 1, \ldots, n-1$. The entries of the right-hand sides were random numbers distributed uniformly in $[0, 1]$. The logarithms of the relative residuals $\eta$ and the magnitude $M$ are plotted versus $n$ in Figure 4.1.

We observe again residuals much larger than $u$ such that $\eta \approx M \cdot u$. Note that both $M$ and the relative residual increase exponentially with $n$, in agreement with (4.14).

![Figure 4.1. Relative residual and $M$ versus size of matrices that have all parameters $t_k = 1/2$.](image)

The last numerical test we present illustrates that the instability of Algorithm B is not a rare phenomenon. To show this, we performed 1000 random experiments with $100 \times 100$ order one quasiseparable matrices parameterized as in (4.1). The generators of these tests were chosen as follows: $G$ and $H$ in (4.2) consisted of sines and cosines of angles distributed uniformly on $[0, 2\pi]$, entries of $v$ and $e$ were chosen to be uniformly distributed on $[-1, 1]$, as well as the entries of right-hand side vectors $b$. The relative residuals of these random tests are summarized by the histogram in Figure 4.2. We also show in Table 4.1 the maximum and minimum of the magnitude $M$ for the cases in each bar of the histogram. Table 4.1 shows that when Algorithm B produces large relative residuals, then large values of $M$ necessarily occur. However, the opposite is not true, because large values of $M$ may
occur and the corresponding relative residuals be small (see the data for the bar of residual e-16). This is not in contradiction with our discussion in Subsection 4.1, since we only showed that $M$ contributes to an upper bound of the backward error.

![Relative residual distribution in 1000 numerical experiments with 100 \times 100 randomly generated matrices.](image)

**Figure 4.2.** Relative residual distribution in 1000 numerical experiments with 100 \times 100 randomly generated matrices.

**Table 4.1.** Maximum and minimum of magnitudes $M$ for each bar of the histogram in Figure 4.2.

<table>
<thead>
<tr>
<th>$\eta$</th>
<th>e+0</th>
<th>e-1</th>
<th>e-2</th>
<th>e-3</th>
<th>e-4</th>
<th>e-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{\text{min}}$</td>
<td>-</td>
<td>7.8e+15</td>
<td>-</td>
<td>1.2e+15</td>
<td>1.0e+13</td>
<td>1.3e+11</td>
</tr>
<tr>
<td>$M_{\text{max}}$</td>
<td>-</td>
<td>2.6e+18</td>
<td>-</td>
<td>6.0e+15</td>
<td>3.2e+13</td>
<td>1.4e+14</td>
</tr>
<tr>
<td>$\eta$</td>
<td>e-6</td>
<td>e-7</td>
<td>e-8</td>
<td>e-9</td>
<td>e-10</td>
<td>e-11</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>$M_{\text{min}}$</td>
<td>2.3e+11</td>
<td>6.3e+9</td>
<td>5.8e+8</td>
<td>5.1e+7</td>
<td>3.2e+6</td>
<td>7.1e+5</td>
</tr>
<tr>
<td>$M_{\text{max}}$</td>
<td>9.2e+13</td>
<td>4.2e+13</td>
<td>4.3e+12</td>
<td>3.6e+12</td>
<td>1.2e+11</td>
<td>2.4e+11</td>
</tr>
<tr>
<td>$\eta$</td>
<td>e-12</td>
<td>e-13</td>
<td>e-14</td>
<td>e-15</td>
<td>e-16</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td>$M_{\text{min}}$</td>
<td>5.1e+4</td>
<td>4.1e+3</td>
<td>4.4e+2</td>
<td>1.0e+2</td>
<td>1.3e+1</td>
<td></td>
</tr>
<tr>
<td>$M_{\text{max}}$</td>
<td>4.7e+9</td>
<td>1.8e+9</td>
<td>1.3e+8</td>
<td>2.8e+9</td>
<td>5.2e+6</td>
<td></td>
</tr>
</tbody>
</table>

5. **Conclusions and future work**

We have started in this paper the development of rounding error analysis of fast algorithms for quasiseparable matrices by analyzing the two fast quasiseparable linear solvers via a structured QR factorization that are presently available in the literature. In this work, we have focused on the subclass of order one quasiseparable
matrices defined by equations (1.1), i.e., those matrices whose diagonal is included in the lower triangular rank structure. This class includes matrices that are very relevant in applications as, for instance, the semiseparable matrices. Despite this simplification, the rounding error analysis we have developed is far from trivial and requires the use of novel approaches that employ carefully the quasiseparable structure. This error analysis proves rigorously that the fast QR-based linear solver introduced in [8] is backward stable on the matrices defined by (1.1), which constitutes the most relevant result in this work. We have also studied carefully the application on matrices (1.1) of the QR-based linear solver presented in [12, 14], and we have constructed explicit counterexamples that show that this algorithm is not backward stable. The sources of instability of the algorithm in [12, 14] have been identified and presented in an understandable way. Besides, we have performed many random numerical tests that show that the appearance of these instabilities in practice is not rare. Two fundamental consequences of this work are: (i) users should employ with caution fast algorithms for quasiseparable matrices since they may be unstable; and (ii) a lot of work remains to be done to identify which fast algorithms for quasiseparable matrices are backward stable among the large family available in the literature. Future work includes the generalization of the error analysis we have developed from matrices defined by (1.1) to general order one quasiseparable matrices and the error analysis of the fast linear solver presented in [2].

References


Instituto de Ciencias Matemáticas CSIC-UAM-UC3M-UCM and Departamento de Matemáticas, Universidad Carlos III de Madrid, Avenida de la Universidad 30, 28911, Leganés, Madrid, Spain

E-mail address: dopico@math.uc3m.es

Department of Mathematics, 196 Auditorium Road, University of Connecticut, Storrs, Connecticut 06269

E-mail address: olshevsky@uconn.edu

School of Mathematics, The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom

E-mail address: P.Zhlobich@ed.ac.uk