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.

Key words. Quasiseparable, semiseparable, QR-decomposition, backward error analysis.

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

Received by the editor May 30, 2011.

2010 *Mathematics Subject Classification.* 65F05, 65G50, 15A06, 15A23, 15B99.

The work of F. M. Dopico was partially supported by the Ministerio de Economía y Competitividad of Spain through the research grant MTM-2009-09281.

This research was partially done while V. Olshevsky held a position as "Catedrático de Excelencia" at Universidad Carlos III de Madrid in the academic year 2009-10.

This research was partially done while P. Zhlobich visited the Department of Mathematics of Universidad Carlos III de Madrid in January-March 2010, partially funded by the Ministerio de Economía y Competitividad of Spain through the grant MTM-2009-09281.

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 an 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 `Algorithms A` and `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 `Algorithms A` and `B` 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

$$\max_{1\le i\le n-1} \operatorname{rank}(A(i:n,1:i)) = \max_{1\le i\le n-1} \operatorname{rank}(A(1:i,i+1:n)) = 1, \tag{1.1}$$

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

$$\left[\begin{array}{c|ccc} \times & \times & \times & \times \\ \times & \times & \multicolumn{1}{|c}{\times} & \times \\ \times & \times & \times & \multicolumn{1}{|c}{\times} \\ \times & \times & \times & \times \end{array}\right],$$

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 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 specially 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.

[1]According to [14] semiseparable matrices are the ones satisfying $\max_i \operatorname{rank}(A(i:n,1:i)) = \max_i \operatorname{rank}(A(1:i,i:n)) = 1$. These matrices are called by other authors *Green's matrices*.

We finally mention that `Algorithm A` has produced always 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 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 $A = QR$ [11], 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 $Rx = Q^T b$, that can be computed by backward substitution [11]. If A satisfies (1.1), then Q and R have particular structures that allow us to compute in $\mathcal{O}(n)$ flops: (i) Q and R; (ii) $Q^T b$; and (iii) the solution of $Rx = Q^T b$. This is much faster than traditional unstructured methods that cost: $\mathcal{O}(n^3)$ for (i), $\mathcal{O}(n^2)$ for (ii), and $\mathcal{O}(n^2)$ 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 $A = QR$ 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,*

$$Q = (W_1 W_2 \cdots W_{n-1})^T, \qquad \operatorname{rank}(R(1:i, i:n)) \leq 2, \qquad 1 \leq i \leq n,$$

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

$$W_k = \begin{bmatrix} I_{k-1} & & & \\ & c_k & s_k & \\ & -s_k & c_k & \\ & & & I_{n-k-1} \end{bmatrix}. \tag{2.1}$$

In the rest of this section, we explain how the rank structure of R can be used to solve in $\mathcal{O}(n)$ 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

$$\max_{1\le i\le n-1} \operatorname{rank}(R(1:i, i+1:n)) = m, \tag{2.2}$$

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 [7, 14, 1]) that every upper triangular quasiseparable matrix of order m, i.e., satisfying (2.2), admits the following representation:

$$R = \begin{bmatrix} d_1 & g_1h_2 & g_1b_2h_3 & \cdots & g_1b_2\dots b_{n-1}h_n \\ 0 & d_2 & g_2h_3 & \cdots & g_2b_3\dots b_{n-1}h_n \\ 0 & 0 & d_3 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & g_{n-1}h_n \\ 0 & 0 & \cdots & 0 & d_n \end{bmatrix}, \tag{2.3}$$

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

	d_k	g_k	b_k	h_k
size	1×1	$1\times m$	$m\times m$	$m\times 1$

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 5×5 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,

$$\begin{aligned} x_5 &= (z_5)/d_5, & \tau_4 &= h_5x_5, \\ x_4 &= (z_4 - g_4\tau_4)/d_4, & \tau_3 &= h_4x_4 + b_4\tau_4, \\ x_3 &= (z_3 - g_3\tau_3)/d_3, & \tau_2 &= h_3x_3 + b_3\tau_3, \\ x_2 &= (z_2 - g_2\tau_2)/d_2, & \tau_1 &= h_2x_2 + b_2\tau_2, \\ x_1 &= (z_1 - g_1\tau_1)/d_1, \end{aligned} \tag{2.4}$$

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

Algorithm 1 Fast quasiseparable backward substitution to solve $Rx = z$.

```
Input: $\{z_k\}_{k=1}^n$ and generators $\{d_k\}_{k=1}^n, \{g_k\}_{k=1}^{n-1}, \{b_k\}_{k=2}^{n-1}, \{h_k\}_{k=2}^n$ of $R$ in (2.3)
  $x_n = z_n/d_n, \quad \tau = x_n h_n$
  for $k = n-1$ to 2 do
    $x_k = (z_k - g_k \tau)/d_k, \quad \tau = x_k h_k + b_k \tau$
  end for
  $x_1 = (z_1 - g_1 \tau)/d_1$
Output: $x_1, x_2, \dots, x_n$
```

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 [8] 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 3.3 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

(3.1)
$$A = \begin{bmatrix} p_1 q_1 & g_1 h_2 & g_1 b_2 h_3 & \cdots & \cdots & g_1 b_2 \dots b_{n-1} h_n \\ p_2 a_2 q_1 & p_2 q_2 & g_2 h_3 & \cdots & \cdots & g_2 b_3 \dots b_{n-1} h_n \\ p_3 a_3 a_2 q_1 & p_3 a_3 q_2 & p_3 q_3 & \cdots & \cdots & g_3 b_4 \dots b_{n-1} h_n \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & p_{n-1} q_{n-1} & g_{n-1} h_n \\ p_n a_n \dots a_2 q_1 & p_n a_n \dots a_3 q_2 & p_n a_n \dots a_4 q_3 & \cdots & p_n a_n q_{n-1} & p_n q_n \end{bmatrix},$$

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 [8], 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 1×2, 2×2, and 2×1, 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:

(3.2)

$$A = \begin{bmatrix} \widetilde{g}_0\widetilde{h}_1 & \widetilde{g}_0\widetilde{b}_1\widetilde{h}_2 & \widetilde{g}_0\widetilde{b}_1\widetilde{b}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_0\widetilde{b}_1\dots\widetilde{b}_{n-1}\widetilde{h}_n \\ p_2a_2q_1 & \widetilde{g}_1\widetilde{h}_2 & \widetilde{g}_1\widetilde{b}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_1\widetilde{b}_2\dots\widetilde{b}_{n-1}\widetilde{h}_n \\ p_3a_3a_2q_1 & p_3a_3q_2 & \widetilde{g}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_2\widetilde{b}_3\dots\widetilde{b}_{n-1}\widetilde{h}_n \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \widetilde{g}_{n-2}\widetilde{h}_{n-1} & \widetilde{g}_{n-2}\widetilde{b}_{n-1}\widetilde{h}_n \\ p_na_n\dots a_2q_1 & p_na_n\dots a_3q_2 & p_na_n\dots a_4q_3 & \cdots & p_na_nq_{n-1} & \widetilde{g}_{n-1}\widetilde{h}_n \end{bmatrix},$$

where

$$\widetilde{g}_k = \begin{bmatrix} 0 & 1 \end{bmatrix}, \quad k = 0, 1, \dots, n-1;$$
$$\widetilde{b}_1 = \begin{bmatrix} 0 & 0 \\ g_1 & 0 \end{bmatrix}, \quad \widetilde{b}_k = \begin{bmatrix} b_k & 0 \\ g_k & 0 \end{bmatrix}, \quad k = 2, \dots, n-1;$$
$$\widetilde{h}_1 = \begin{bmatrix} 0 \\ p_1q_1 \end{bmatrix}, \quad \widetilde{h}_k = \begin{bmatrix} h_k \\ p_kq_k \end{bmatrix}, \quad k = 2, \dots, n.$$

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:

$$\bar{p}_n = p_n, \quad \bar{p}_k = \sqrt{(\bar{p}_{k+1}a_{k+1})^2 + p_k^2}, \quad k = n-1, \dots 1,$$
$$G_k = \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}, \quad s_k = \bar{p}_{k+1}a_{k+1}/\bar{p}_k, \quad c_k = p_k/\bar{p}_k,$$

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 $\bar{p}_k \neq 0$, for $k = n, n-1, \dots, 1$.

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 $\{\widetilde{g}_k, \widetilde{b}_k, \widetilde{h}_k\}$ according to (3.2). The resulting matrix R is also described by some generators $\{\bar{g}_k, \bar{b}_k, \bar{h}_k\}$ with the same sizes, i.e., 1×2, 2×2, and 2×1 respectively. So, the next idea of `Algorithm 2` is to find direct computational formulas to relate both sets of parameters

$$\{\widetilde{g}_k\}_{k=0}^{n-1}, \{\widetilde{b}_k\}_{k=1}^{n-1}, \{\widetilde{h}_k\}_{k=1}^{n} \longrightarrow \{\bar{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^{n},$$

that are mathematically equivalent to Givens rotations. To deduce these formulas, let us assume for $k > 1$ that at every step $n-1, n-2, \dots, k+1$, introducing zeros in rows $n, n-1, \dots, k+2$, respectively, the algorithm modified in the upper triangular part only the generators with indices $n-1, n-2, \dots, k+1$. 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

$$\begin{bmatrix} \widetilde{g}_{k-1}\widetilde{h}_k & \widetilde{g}_{k-1}\widetilde{b}_k\bar{h}_{k+1} & \widetilde{g}_{k-1}\widetilde{b}_k\bar{b}_{k+1}\bar{h}_{k+2} & \cdots & \widetilde{g}_{k-1}\widetilde{b}_k\cdots\bar{b}_{n-1}\bar{h}_n \\ \bar{p}_{k+1}a_{k+1}q_k & \widetilde{g}_k\bar{h}_{k+1} & \widetilde{g}_k\bar{b}_{k+1}\bar{h}_{k+2} & \cdots & \widetilde{g}_k\bar{b}_{k+1}\dots\bar{b}_{n-1}\bar{h}_n \end{bmatrix} =$$
$$= \begin{bmatrix} \widetilde{g}_{k-1}\widetilde{h}_k & \widetilde{g}_{k-1}\widetilde{b}_k \\ \bar{p}_{k+1}a_{k+1}q_k & \widetilde{g}_k \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \bar{h}_{k+1} & \bar{b}_{k+1}\bar{h}_{k+2} & \cdots & \bar{b}_{k+1}\dots\bar{b}_{n-1}\bar{h}_n \end{bmatrix}.$$

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

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

where we introduced the transformed generators

$$\bar{g}_k := \begin{bmatrix} -s_k g_k & c_k \end{bmatrix}, \quad \bar{b}_k := \begin{bmatrix} b_k & 0 \\ c_k g_k & s_k \end{bmatrix}, \quad \bar{h}_k := \begin{bmatrix} h_k \\ \bar{p}_k q_k \end{bmatrix}. \tag{3.3}$$

Note that the transformation $\{\widetilde{g}_k, \widetilde{b}_k, \widetilde{h}_k\} \longrightarrow \{\bar{g}_k, \bar{b}_k, \bar{h}_k\}$ only changes the rows k and $k+1$ and does not affect any other rows of $A^{(k)}$, despite the fact that $\bar{b}_k$ and $\bar{h}_k$ also appear in rows $1, \dots, k-1$. 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)
  $\bar{p}_n = p_n, \quad \bar{h}_n = \begin{bmatrix} h_n \\ p_n q_n \end{bmatrix}$
  for $k = n-1$ to $2$ do
    $\bar{p}_k = \sqrt{(\bar{p}_{k+1}a_{k+1})^2 + p_k^2}, \quad s_k = \bar{p}_{k+1}a_{k+1}/\bar{p}_k, \quad c_k = p_k/\bar{p}_k$
    $\bar{g}_k = \begin{bmatrix} -s_k g_k & c_k \end{bmatrix}, \quad \bar{b}_k = \begin{bmatrix} b_k & 0 \\ c_k g_k & s_k \end{bmatrix}, \quad \bar{h}_k = \begin{bmatrix} h_k \\ \bar{p}_k q_k \end{bmatrix}$
  end for
  $\bar{p}_1 = \sqrt{(\bar{p}_2 a_2)^2 + p_1^2}, \quad s_1 = \bar{p}_2 a_2/\bar{p}_1, \quad c_1 = p_1/\bar{p}_1$
  $\bar{g}_1 = \begin{bmatrix} -s_1 g_1 & c_1 \end{bmatrix}, \quad \bar{b}_1 = \begin{bmatrix} 0 & 0 \\ c_1 g_1 & s_1 \end{bmatrix}, \quad \bar{h}_1 = \begin{bmatrix} 0 \\ \bar{p}_1 q_1 \end{bmatrix}$
  $\bar{g}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}$
Output: $\{(c_k, s_k)\}_{k=1}^{n-1}$ cosines and sines of Givens rotations $W_k$ in (2.1) and gen-
    erators of the factor $R$ of $A$, $\{\bar{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^n$, indexed as in (3.4).
```

Note that the expressions of the generators $\bar{b}_1$ and $\bar{h}_1$ 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 $\bar{b}_1$ and $\bar{h}_1$ only appear in the first row of R. We invite the reader to check that the formulas for $\bar{b}_1$ and $\bar{h}_1$ 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* $\{\bar{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^n$

be the generators computed by `Algorithm 2`*. Assume that all computations are performed in exact arithmetic. Then the matrix*

$$R = \begin{bmatrix} \bar g_0 \bar h_1 & \bar g_0 \bar b_1 \bar h_2 & \bar g_0 \bar b_1 \bar b_2 \bar h_3 & \cdots & \cdots & \bar g_0 \bar b_1 \ldots \bar b_{n-1} \bar h_n \\ 0 & \bar g_1 \bar h_2 & \bar g_1 \bar b_2 \bar h_3 & \cdots & \cdots & \bar g_1 \bar b_2 \ldots \bar b_{n-1} \bar h_n \\ 0 & 0 & \bar g_2 \bar h_3 & \cdots & \cdots & \bar g_2 \bar b_3 \ldots \bar b_{n-1} \bar h_n \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \bar g_{n-2} \bar h_{n-1} & \bar g_{n-2} \bar b_{n-1} \bar h_n \\ 0 & 0 & 0 & \cdots & 0 & \bar g_{n-1} \bar h_n \end{bmatrix}, \tag{3.4}$$

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 (3.4). This adaptation yields `Algorithm 3`, which is valid for any set of generators $\{\bar g_k\}_{k=0}^{n-1}$,$\{\bar b_k\}_{k=1}^{n-1}$,$\{\bar h_k\}_{k=1}^{n}$ of sizes 1×2, 2×2, and 2×1, 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:

Generators in (2.3)	**Generators in** (3.4)
$\{d_1, d_2, \ldots, d_n\}$	$\{\bar g_0 \bar h_1, \bar g_1 \bar h_2, \ldots, \bar g_{n-1} \bar h_n\}$
$\{h_2, h_3, \ldots, h_n\}$	$\{\bar b_1 \bar h_2, \bar b_2 \bar h_3, \ldots, \bar b_{n-1} \bar h_n\}$
$\{b_2, b_3, \ldots, b_{n-1}\}$	$\{\bar b_1, \bar b_2, \ldots, \bar b_{n-2}\}$
$\{g_1, g_2, \ldots, g_{n-1}\}$	$\{\bar g_0, \bar g_1, \ldots, \bar g_{n-2}\}$

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 $\{\bar g_k\}_{k=0}^{n-1}$, $\{\bar b_k\}_{k=1}^{n-1}$, $\{\bar h_k\}_{k=1}^{n}$ of $R$ in (3.4)
  for $k = 1$ to $n$ do
    $d_k = \bar g_{k-1} \bar h_k$
  end for
  $x_n = z_n / d_n, \quad \tau = x_n\, (\bar b_{n-1} \bar h_n)$
  for $k = n-1$ to $2$ do
    $x_k = (z_k - \bar g_{k-1} \tau)/d_k, \quad \tau = \bar b_{k-1}\, (x_k \bar h_k + \tau)$
  end for
  $x_1 = (z_1 - \bar g_0 \tau)/d_1$
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 $\{W_k\}_{k=1}^{n-1}$ and generators of the factor $R$ of $A$, $\{\bar{g}_k\}_{k=0}^{n-1}, \{\bar{b}_k\}_{k=1}^{n-1}, \{\bar{h}_k\}_{k=1}^n$
  Step 2: $z = (W_1 \dots (W_{n-2}(W_{n-1}b)) \dots)$
  Step 3: Apply Algorithm 3 to solve $Rx = z$ and compute $x$
Output: $x_1, x_2, \dots, x_n$
```

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

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta) = \frac{x \operatorname{op} y}{1+\alpha}, \qquad |\delta|, |\alpha| \le \mathbf{u}, \quad \operatorname{op} = +, -, *, /, \tag{3.5}$$
$$fl(\sqrt{x}) = \sqrt{x}(1+\varepsilon), \qquad |\varepsilon| \le \mathbf{u},$$

where $\mathbf{u}$ 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 $|\delta_i| \le \mathbf{u}$ and $\rho_i = \pm 1$ for $i = 1, \dots, k$, and $k\mathbf{u} < 1$, then

$$\prod_{i=1}^{k} (1+\delta_i)^{\rho_i} = 1 + \Theta_k, \quad \text{where} \quad |\Theta_k| \le \frac{k\mathbf{u}}{1-k\mathbf{u}} =: \gamma_k$$

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 $B \le C$ mean $b_{ij} \le c_{ij}$ 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 $R_{\widehat{g}}$ 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 column-wise backward errors in A are preserved if $R_{\widehat{g}}$ is perturbed via tiny componentwise variations, *that may be different for each column of* $R_{\widehat{g}}$, of the computed generators. Part (b) is the key to match the backward errors in `Algorithm 3` with those of `Algorithm 2`.

Theorem 3.2. *Let* $\{\widehat{g}_k\}_{k=0}^{n-1}$, $\{\widehat{\overline{b}}_k\}_{k=1}^{n-1}$, *and* $\{\widehat{\overline{h}}_k\}_{k=1}^{n}$ *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 $R_{\widehat{g}}$ be the upper triangular matrix exactly constructed from* $\{\widehat{g}_k\}_{k=0}^{n-1}$, $\{\widehat{\overline{b}}_k\}_{k=1}^{n-1}$, $\{\widehat{\overline{h}}_k\}_{k=1}^{n}$ *according to* (3.4)*. Then there exists an exact orthogonal $n \times n$ matrix Q such that*

$$A + E = QR_{\widehat{g}}, \quad \textit{where} \quad \|E(:,j)\|_2 \leq \gamma_{C_1 n}\|A(:,j)\|_2, \;\; j = 1,\dots,n,$$

and C_1 is a small integer constant. The matrix Q is given explicitly as $Q = (\mathcal{W}_1\mathcal{W}_2\cdots\mathcal{W}_{n-1})^T$, where $\mathcal{W}_k$ is an exact Givens rotation with the structure given in (2.1) *and based on the exact sine and cosine*

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

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

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

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

where, for a certain nonnegative integer p,

$$\begin{aligned} |\bar{g}_i^{(j)} - \widehat{g}_i| &\leq \gamma_{\mathrm{p}}|\widehat{g}_i| \quad \textit{for} \quad 1 \leq j \leq n,\; 0 \leq i \leq j-1, \\ |\bar{b}_i^{(j)} - \widehat{\overline{b}}_i| &\leq \gamma_{\mathrm{p}}|\widehat{\overline{b}}_i| \quad \textit{for} \quad 1 \leq j \leq n,\; 1 \leq i \leq j-1, \\ |\bar{h}_j^{(j)} - \widehat{\overline{h}}_j| &\leq \gamma_{\mathrm{p}}|\widehat{\overline{h}}_j| \quad \textit{for} \quad 1 \leq j \leq n. \end{aligned}$$

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

$$A + F = QR_{\widehat{g}+\delta g}, \;\; \textit{where} \; \|F(:,j)\|_2 \leq \gamma_{C_1 n + C_2 \mathrm{p} n^2}\|A(:,j)\|_2, \quad j = 1,\dots,n,$$

and C_2 is a small integer constant.

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

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

$$\begin{aligned}
&\widehat{s}_i = fl\left(\widehat{p}_{i+1}a_{i+1}/\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2}\right) = s_i'(1+\Theta_6),\\
&\text{where } s_i' = \widehat{p}_{i+1}a_{i+1}/\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2},\\
&\widehat{c}_i = fl\left(p_i/\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2}\right) = c_i'(1+\Theta_5),\\
&\text{where } c_i' = p_i/\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2}.
\end{aligned} \tag{3.6}$$

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 column-wise 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 in two steps.

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

$$\begin{aligned}
\widehat{p}_i &= fl\left(\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2}\right) = (1+\Theta_4)\sqrt{(\widehat{p}_{i+1}a_{i+1})^2 + p_i^2}\\
&= \begin{bmatrix} c_i'(1+\Theta_4) & s_i'(1+\Theta_4) \end{bmatrix} \begin{bmatrix} p_i \\ \widehat{p}_{i+1}a_{i+1} \end{bmatrix}.
\end{aligned}$$

Hence,

$$\begin{bmatrix} \widehat{p}_i \\ 0 \end{bmatrix} = \begin{bmatrix} c_i'(1+\Theta_4) & s_i'(1+\Theta_4) \\ -s_i' & c_i' \end{bmatrix} \begin{bmatrix} p_i \\ \widehat{p}_{i+1}a_{i+1} \end{bmatrix} = (G_i + \Delta G_i) \begin{bmatrix} p_i \\ \widehat{p}_{i+1}a_{i+1} \end{bmatrix}, \tag{3.7}$$

where

$$G_i := \begin{bmatrix} c_i' & s_i' \\ -s_i' & c_i' \end{bmatrix}, \qquad \text{and} \qquad \Delta G_i := \begin{bmatrix} c_i'\Theta_4 & s_i'\Theta_4 \\ 0 & 0 \end{bmatrix}. \tag{3.8}$$

In order to simplify the notation, here and thereafter let us agree to denote by G_i a 2×2 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 ΔG_i, with the additional property that all entries of ΔG_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 $a_i \dots a_{j+1}q_j$ in exact arithmetic we get:

$$\begin{bmatrix} \widehat{p}_i a_i \dots a_{j+1}q_j \\ 0 \end{bmatrix} = (G_i + \Delta G_i) \begin{bmatrix} p_i a_i \dots a_{j+1}q_j \\ \widehat{p}_{i+1}a_{i+1}a_i \dots a_{j+1}q_j \end{bmatrix},$$

which, by induction, leads to

$$\begin{bmatrix} \widehat{\overline{p}}_j q_j \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \prod_{i=j}^{n-1} (G_i + \Delta G_i) \begin{bmatrix} p_j q_j \\ p_{j+1} a_{j+1} q_j \\ \vdots \\ p_n a_n \dots a_{j+1} q_j \end{bmatrix} = \prod_{i=j}^{n-1} (G_i + \Delta G_i) A(j:n, j). \tag{3.9}$$

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 $j, j-1, \dots, 2$. Let us look at the computed generators in `Algorithm 2` that contribute to column $R_{\widehat{g}}(:,j)$:

$$\begin{aligned} \widehat{\overline{g}}_i &= \begin{bmatrix} -\widehat{s}_i g_i (1+\Theta_1) & \widehat{c}_i \end{bmatrix} = \begin{bmatrix} -s_i' g_i (1+\Theta_7) & c_i'(1+\Theta_5) \end{bmatrix}, & 0 \le i \le j-1, \\ \widehat{\overline{b}}_i &= \begin{bmatrix} b_i & 0 \\ \widehat{c}_i g_i (1+\Theta_1') & \widehat{s}_i \end{bmatrix} = \begin{bmatrix} b_i & 0 \\ c_i' g_i (1+\Theta_6) & s_i'(1+\Theta_6') \end{bmatrix}, & 1 \le i \le j-1 \\ \widehat{\overline{h}}_j &= \begin{bmatrix} h_j \\ \widehat{\overline{p}}_j q_j (1+\Theta_1'') \end{bmatrix}, \end{aligned}$$

where, for simplicity, we define $g_0 = \widehat{s}_0 = s_0' = b_1 = h_1 = 0$ and $\widehat{c}_0 = c_0' = 1$ to take into account the special structure of generators $\overline{g}_0, \overline{b}_1$ and $\overline{h}_1$. Note also that the Θ_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 $\overline{g}_i^{(j)}$, $\overline{b}_i^{(j)}$ and $\overline{h}_j^{(j)}$ in the statement of part (b) of Theorem 3.2, they are obtained from $\widehat{\overline{g}}_i$, $\widehat{\overline{b}}_i$, and $\widehat{\overline{h}}_j$ by introducing tiny componentwise perturbations bounded by γ_{p}. Therefore, we can write

$$\begin{aligned} \overline{g}_i^{(j)} &= \begin{bmatrix} -s_i' g_i (1+\Theta_{\mathrm{p}+7}) & c_i'(1+\Theta_{\mathrm{p}+5}) \end{bmatrix}, & 0 \le i \le j-1 \\ \overline{b}_i^{(j)} &= \begin{bmatrix} b_i(1+\Theta_{\mathrm{p}}) & 0 \\ c_i' g_i (1+\Theta_{\mathrm{p}+6}) & s_i'(1+\Theta_{\mathrm{p}+6}') \end{bmatrix}, & 1 \le i \le j-1 \\ \overline{h}_j^{(j)} &= \begin{bmatrix} h_j(1+\Theta_{\mathrm{p}}') \\ \widehat{\overline{p}}_j q_j (1+\Theta_{\mathrm{p}+1}) \end{bmatrix}. \end{aligned} \tag{3.10}$$

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

$$\widetilde{r}_{ij} := (R_{\widehat{g}+\delta g})_{ij} = \overline{g}_{i-1}^{(j)} \overline{b}_i^{(j)} \cdots \overline{b}_{j-1}^{(j)} \overline{h}_j^{(j)}, \qquad \text{if } i < j,$$

and $\widetilde{r}_{jj} := (R_{\widehat{g}+\delta g})_{jj} = \overline{g}_{j-1}^{(j)} \overline{h}_j^{(j)}$. Note that all the entries $\widetilde{r}_{ij}$, for $i = j, \dots, 1$, can be computed by multiplying generators from the end to the beginning as follows:

$$\begin{gathered} \tau_j = \overline{h}_j^{(j)}, \\ \widetilde{r}_{ij} = \overline{g}_{i-1}^{(j)} \tau_i, \quad \tau_{i-1} = \overline{b}_{i-1}^{(j)} \tau_i, \quad i = j, \dots 2, \\ \widetilde{r}_{1j} = \overline{g}_0^{(j)} \tau_1, \end{gathered} \tag{3.11}$$

here $\tau_i = \begin{bmatrix} \tau_{i,1} & \tau_{i,2} \end{bmatrix}^T$ is an auxiliary column vector of length two.

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

$$\begin{bmatrix} \tau_{i-1,2} \\ \widetilde{r}_{ij} \end{bmatrix} = \begin{bmatrix} c'_{i-1}(1+\Theta_{\mathrm{p}+6}) & s'_{i-1}(1+\Theta'_{\mathrm{p}+6}) \\ -s'_{i-1}(1+\Theta_{\mathrm{p}+7}) & c'_{i-1}(1+\Theta_{\mathrm{p}+5}) \end{bmatrix} \begin{bmatrix} g_{i-1}\tau_{i,1} \\ \tau_{i,2} \end{bmatrix}, \tag{3.12}$$
$$\tau_{i-1,1} = b_{i-1}\tau_{i,1}(1+\Theta_{\mathrm{p}}).$$

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

$$\begin{bmatrix} \tau_{j-1,2} \\ \widetilde{r}_{jj} \end{bmatrix} = \begin{bmatrix} c'_{j-1}(1+\Theta_{\mathrm{p}+6}) & s'_{j-1}(1+\Theta'_{\mathrm{p}+6}) \\ -s'_{j-1}(1+\Theta_{\mathrm{p}+7}) & c'_{j-1}(1+\Theta_{\mathrm{p}+5}) \end{bmatrix} \begin{bmatrix} g_{j-1}h_j(1+\Theta'_{\mathrm{p}}) \\ \widehat{\widehat{p}}_j q_j(1+\Theta_{\mathrm{p}+1}) \end{bmatrix},$$
$$\tau_{j-1,1} = b_{j-1}h_j(1+\Theta_{2\mathrm{p}}),$$

or, attaching errors to the Givens rotation,

$$\begin{bmatrix} \tau_{j-1,2} \\ \widetilde{r}_{jj} \end{bmatrix} = \begin{bmatrix} c'_{j-1}(1+\Theta_{2\mathrm{p}+6}) & s'_{j-1}(1+\Theta_{2\mathrm{p}+7}) \\ -s'_{j-1}(1+\Theta'_{2\mathrm{p}+7}) & c'_{j-1}(1+\Theta'_{2\mathrm{p}+6}) \end{bmatrix} \begin{bmatrix} g_{j-1}h_j \\ \widehat{\widehat{p}}_j q_j \end{bmatrix},$$
$$\tau_{j-1,1} = b_{j-1}h_j(1+\Theta_{2\mathrm{p}}).$$

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

$$\tau_{i,1} = b_i b_{i+1} \cdots b_{j-1} h_j (1+\Theta_{(j-i+1)\mathrm{p}}), \qquad \text{for } i = j-1, \dots, 1.$$

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

$$\begin{bmatrix} \tau_{i-1,2} \\ \widetilde{r}_{ij} \end{bmatrix} = \begin{bmatrix} c'_{i-1}(1+\Theta_{(j-i+2)\mathrm{p}+6}) & s'_{i-1}(1+\Theta_{\mathrm{p}+6}) \\ -s'_{i-1}(1+\Theta_{(j-i+2)\mathrm{p}+7}) & c'_{i-1}(1+\Theta_{\mathrm{p}+5}) \end{bmatrix} \begin{bmatrix} g_{i-1}b_i \dots b_{j-1}h_j \\ \tau_{i,2} \end{bmatrix}. \tag{3.13}$$

Recall the simplified notation introduced in (3.8) and the paragraph just below it[2], and stick equations (3.13) together for $i = 2, \dots, j$:

$$\begin{bmatrix} \tau_{1,2} \\ \widetilde{r}_{2j} \\ \vdots \\ \vdots \\ \widetilde{r}_{jj} \end{bmatrix} = \prod_{i=1}^{j-1}(G_i + \Delta G_i) \begin{bmatrix} g_1 b_2 \dots b_{j-1} h_j \\ g_2 b_3 \dots b_{j-1} h_j \\ \vdots \\ g_{j-1} h_j \\ \widehat{\widehat{p}}_j q_j \end{bmatrix} = \prod_{i=1}^{j-1}(G_i + \Delta G_i) \begin{bmatrix} A(1:j-1,j) \\ \widehat{\widehat{p}}_j q_j \end{bmatrix}. \tag{3.14}$$

We still need to relate $\tau_{1,2}$ to $\widetilde{r}_{1j}$. But this is trivial, since (3.11) and $\widehat{g}_0 = \begin{bmatrix} 0 & 1 \end{bmatrix}$ imply $\widetilde{r}_{1j} = (1+\Theta_{\mathrm{p}})\tau_{1,2}$. Therefore, one can replace $\tau_{1,2}$ in (3.14) by $\widetilde{r}_{1j}$ at the cost of attaching the extra error $1+\Theta_{\mathrm{p}}$ to the first row of $G_1 + \Delta G_1$, and so modifying the first row of ΔG_1.

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

$$R_{\widehat{g}+\delta g}(:,j) = \prod_{i=1}^{n-1}(\mathcal{W}_i + \Delta\mathcal{W}_i)A(:,j), \tag{3.15}$$

[2]The reader should observe that in step 2 of the proof, that is, for $i = 1, \dots, j-1$, the backward errors in Givens rotations are $\Delta G_i = [\, c'_i\,\Theta_{(j-i+1)\mathrm{p}+6} \,,\; s'_i\,\Theta_{\mathrm{p}+6} \,;\; -s'_i\,\Theta_{(j-i+1)\mathrm{p}+7} \,,\; c'_i\,\Theta_{\mathrm{p}+5} \,]$.

where $\mathcal{W}_1, \dots, \mathcal{W}_{n-1}$ are the Givens rotations defined in part (a) of the statement of Theorem 3.2. Equations (3.8) and (3.13) define $\Delta\mathcal{W}_i$ and imply that

$$\begin{aligned}
&\|\Delta\mathcal{W}_1\|_F \leq \sqrt{2}\gamma_{(j+1)\mathrm{p}+6} \leq \sqrt{2}\gamma_{(j+1)\mathrm{p}+6}\,\|\mathcal{W}_1\|_2,\\
&\|\Delta\mathcal{W}_i\|_F \leq \sqrt{2}\gamma_{(j-i+1)\mathrm{p}+7} \leq \sqrt{2}\gamma_{(j-i+1)\mathrm{p}+7}\,\|\mathcal{W}_i\|_2, \qquad i = 2, \dots, j-1,\\
&\|\Delta\mathcal{W}_i\|_F \leq \gamma_4 \leq \gamma_4\,\|\mathcal{W}_i\|_2, \qquad i = j, \dots, n-1.
\end{aligned}$$

Therefore, using $Q^T = \prod_{i=1}^{n-1} \mathcal{W}_i$ and [11, Lemma 3.7], we get

$$R_{\widehat{g}+\delta g}(:, j) = \left(\prod_{i=1}^{n-1} \mathcal{W}_i + \prod_{i=1}^{n-1} (\mathcal{W}_i + \Delta\mathcal{W}_i) - \prod_{i=1}^{n-1} \mathcal{W}_i \right) A(:, j) = Q^T(A(:, j) + F(:, j)),$$

with

$$\|F(:, j)\|_2 \leq \left\| \prod_{i=1}^{n-1} (\mathcal{W}_i + \Delta\mathcal{W}_i) - \prod_{i=1}^{n-1} \mathcal{W}_i \right\|_F \|A(:, j)\|_2 \leq \gamma_{C_1 n + C_2 \mathrm{p} n^2} \|A(:, j)\|_2,$$

which proves the result. □

Observe that parts (a) and (b) of Theorem 3.2 imply that $R_{\widehat{g}+\delta g}(:, j) - R_{\widehat{g}}(:, j) = Q^T(F(:, j) - E(:, j))$ and, therefore, $\|R_{\widehat{g}+\delta g}(:, j) - R_{\widehat{g}}(:, j)\|_2 \leq \gamma_{2C_1 n + C_2 \mathrm{p} n^2} \|A(:, j)\|_2$, for $j = 1, \dots, n$. This means that tiny relative componentwise perturbations of the generators of $R_{\widehat{g}}(:, j)$ produce tiny relative variations *in the whole* j-th column, i.e., the generators determine well the columns of $R_{\widehat{g}}$, 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 $R_{\widehat{g}}$ 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 $R_{\widehat{g}}$ 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 $\{\widehat{g}_k\}_{k=0}^{n-1}, \{\widehat{\widehat{b}}_k\}_{k=1}^{n-1}, \{\widehat{\widehat{h}}_k\}_{k=1}^{n}$ 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 $R_{\widehat{g}}$ be the matrices in the statement of Theorem 3.2. Consider the following iteration to compute, for $j = 1, \dots, n$, the entries $\mathring{r}_{jj}, \mathring{r}_{j-1,j}, \dots, \mathring{r}_{1j}$ of $R_{\widehat{g}}$:*

$$\begin{aligned}
&\tau_j = \widehat{\widehat{h}}_j,\\
&\mathring{r}_{ij} = \widehat{\widehat{g}}_{i-1}\tau_i, \quad \tau_{i-1} = \widehat{\widehat{b}}_{i-1}\tau_i, \quad i = j, \dots 2, \qquad (3.16)\\
&\mathring{r}_{1j} = \widehat{\widehat{g}}_0 \tau_1,
\end{aligned}$$

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

$$Q\widehat{R}_{\widehat{g}} = A + \Delta A, \quad \textit{where} \quad \|\Delta A(:, j)\|_2 \leq \gamma_{Cn^2}\|A(:, j)\|_2, \quad j = 1, \dots, n,$$

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 $\widehat{R}_{\widehat{g}}$ introduce componentwise relative backward errors, i.e., perturbations, bounded by γ_2 in generators $\widehat{\overline{g}}_k$ and $\widehat{\overline{b}}_k$. 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 $\widehat{Q}$ is the orthogonal factor of the QR factorization of A computed in floating point arithmetic by multiplying explicitly the Givens rotations, then $\widehat{Q}\widehat{R}_{\widehat{g}} = A + \Delta A$, with ΔA bounded as in Corollary 3.3 with slightly different constants. Moreover, it can be proved that $\|\widehat{R}_{\widehat{g}}(:,j) - R_{\widehat{g}}(:,j)\|_2 \leq \gamma_{C'n^2}\|R_{\widehat{g}}(:,j)\|_2$, for all j. This means that $R_{\widehat{g}}$ 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 1×2, 2×2, and 2×1. The same holds for the error analysis. Observe that the analysis attaches backward errors to the input generators, and that for the generators $\bar{g}_k$ the errors are different for different columns.

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

$$R_{g+\delta g}\widehat{x} = z, \tag{3.17}$$

where the matrix $R_{g+\delta g}$ is defined as

$$R_{g+\delta g} = \begin{bmatrix} \widetilde{\mathbf{g}}_{\mathbf{0}}\widetilde{h}_1 & \widetilde{g}_0\widetilde{b}_1\widetilde{h}_2 & \widetilde{g}_0\widetilde{b}_1\widetilde{b}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_0\widetilde{b}_1\ldots\widetilde{b}_{n-1}\widetilde{h}_n \\ 0 & \widetilde{\mathbf{g}}_1\widetilde{h}_2 & \widetilde{g}_1\widetilde{b}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_1\widetilde{b}_2\ldots\widetilde{b}_{n-1}\widetilde{h}_n \\ 0 & 0 & \widetilde{\mathbf{g}}_2\widetilde{h}_3 & \cdots & \cdots & \widetilde{g}_2\widetilde{b}_3\ldots\widetilde{b}_{n-1}\widetilde{h}_n \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \widetilde{\mathbf{g}}_{n-2}\widetilde{h}_{n-1} & \widetilde{g}_{n-2}\widetilde{b}_{n-1}\widetilde{h}_n \\ 0 & 0 & 0 & \cdots & 0 & \widetilde{\mathbf{g}}_{n-1}\widetilde{h}_n \end{bmatrix} \tag{3.18}$$

in terms of certain generators $\{\widetilde{g}_k\}_{k=0}^{n-2} \subset \mathbb{R}^{1\times 2}$, $\{\widetilde{\mathbf{g}}_k\}_{k=0}^{n-1} \subset \mathbb{R}^{1\times 2}$, $\{\widetilde{b}_k\}_{k=1}^{n-1} \subset \mathbb{R}^{2\times 2}$, and $\{\widetilde{h}_k\}_{k=1}^{n} \subset \mathbb{R}^{2\times 1}$ that satisfy

$$\begin{aligned} |\widetilde{g}_k - \bar{g}_k| &\leq \gamma_2|\bar{g}_k|, \quad \textit{for} \quad k = 0,\ldots,n-2, \\ |\widetilde{\mathbf{g}}_k - \bar{g}_k| &\leq \gamma_5|\bar{g}_k|, \quad \textit{for} \quad k = 0,\ldots,n-1, \\ |\widetilde{b}_k - \bar{b}_k| &\leq \gamma_3|\bar{b}_k|, \quad \textit{for} \quad k = 1,\ldots,n-1, \\ |\widetilde{h}_k - \bar{h}_k| &\leq \mathbf{u}|\bar{h}_k|, \quad \textit{for} \quad k = 1,\ldots,n. \end{aligned} \tag{3.19}$$

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

$$Tx = z, \tag{3.20}$$

where $t_{ij} = r_{ij}$ for $i \neq j$, and $t_{ii} = d_i$ for $i = 1, \ldots, n$.

Let $\widehat{d}_k = fl(\bar{g}_{k-1}\bar{h}_k)$, 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 $\widehat{x}$ satisfies[3]

$$\widehat{x}_n = \frac{z_n}{\widehat{d}_n(1+\varepsilon_n)}, \qquad |\varepsilon_n| \le \mathbf{u},$$
$$\widehat{\tau}_{n-1} = \widehat{x}_n(\bar{b}_{n-1} + \Delta\bar{b}_{n-1})\bar{h}_n, \qquad |\Delta\bar{b}_{n-1}| \le \gamma_3\, |\bar{b}_{n-1}|,$$

```
for $k = n-1$ to 2 do
    $\widehat{x}_k = \frac{z_k - (\bar{g}_{k-1} + \Delta\bar{g}_{k-1})\widehat{\tau}_k}{\widehat{d}_k(1+\varepsilon_k)(1+\varepsilon'_k)}, \quad |\varepsilon_k|, |\varepsilon'_k| \le \mathbf{u}, \quad |\Delta\bar{g}_{k-1}| \le \gamma_2\, |\bar{g}_{k-1}|,$
    $\widehat{\tau}_{k-1} = (\bar{b}_{k-1} + \Delta\bar{b}_{k-1})(\widehat{x}_k(\bar{h}_k + \Delta\bar{h}_k) + \widehat{\tau}_k), \quad |\Delta\bar{b}_{k-1}| \le \gamma_3\, |\bar{b}_{k-1}|, \quad |\Delta\bar{h}_k| \le \mathbf{u}\, |\bar{h}_k|,$
end for
```

$$\widehat{x}_1 = \frac{z_1 - (\bar{g}_0 + \Delta\bar{g}_0)\widehat{\tau}_1}{\widehat{d}_1(1+\varepsilon_1)(1+\varepsilon'_1)}, \qquad |\varepsilon_1|, |\varepsilon'_1| \le \mathbf{u}, \quad |\Delta\bar{g}_0| \le \gamma_2\, |\bar{g}_0|.$$

Comparing this recurrence with `Algorithm 3`, recalling (3.20), and defining $\varepsilon'_n := 0$, we get that $\widehat{x}$ is the exact solution of

$$R_{g+\delta g}\widehat{x} = z,$$

where the entries of $R_{g+\delta g}$ are given by

$$(R_{g+\delta g})_{ij} = \begin{cases} \widehat{d}_i(1+\varepsilon_i)(1+\varepsilon'_i) & \text{if } i = j, \\ (\bar{g}_{i-1} + \Delta\bar{g}_{i-1})(\bar{b}_i + \Delta\bar{b}_i)\cdots(\bar{b}_{j-1} + \Delta\bar{b}_{j-1})(\bar{h}_j + \Delta\bar{h}_j) & \text{if } i < j, \\ 0 & \text{if } i > j, \end{cases}.$$

To finish the proof of Theorem 3.4 we first define $\widetilde{g}_i := \bar{g}_i + \Delta\bar{g}_i$, for $i = 0, \ldots, n-2$, $\widetilde{b}_i = \bar{b}_i + \Delta\bar{b}_i$, for $i = 1, \ldots, n-1$, and $\widetilde{h}_i = \bar{h}_i + \Delta\bar{h}_i$, for $i = 1, \ldots, n$ (we take $\Delta\bar{h}_n = \Delta\bar{h}_1 = 0$). Second, note that, for $k = 1, \ldots, n$, if $\bar{g}_{k-1} = \begin{bmatrix} \bar{g}^1_{k-1} & \bar{g}^2_{k-1} \end{bmatrix}$ then

$$\widehat{d}_k = \begin{bmatrix} \bar{g}^1_{k-1}\,(1+\Theta_2) & \bar{g}^2_{k-1}\,(1+\Theta'_2) \end{bmatrix} \bar{h}_k. \tag{3.21}$$

In addition,

$$\widetilde{h}_k = \begin{bmatrix} (1+\delta_1) & 0 \\ 0 & (1+\delta_2) \end{bmatrix} \bar{h}_k,$$

with $|\delta_1| \le \mathbf{u}$, $|\delta_2| \le \mathbf{u}$. From (3.21), we get $\widehat{d}_k = \begin{bmatrix} \bar{g}^1_{k-1}\,\frac{(1+\Theta_2)}{(1+\delta_1)} & \bar{g}^2_{k-1}\,\frac{(1+\Theta'_2)}{(1+\delta_2)} \end{bmatrix} \widetilde{h}_k$, and

$$(R_{g+\delta g})_{kk} = \widehat{d}_k(1+\varepsilon_k)(1+\varepsilon'_k) = \begin{bmatrix} \bar{g}^1_{k-1}\,(1+\Theta_5) & \bar{g}^2_{k-1}\,(1+\Theta'_5) \end{bmatrix} \widetilde{h}_k \equiv \widetilde{\mathbf{g}}_{k-1}\widetilde{h}_k.$$

This completes the proof. □

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

Theorem 3.5 (`Algorithm A` **is backward stable**). *Let A be an $n \times n$ quasiseparable matrix of type* (1.1) *given by its generators* $\{p_k\}_{k=1}^n$, $\{a_k\}_{k=2}^n$, $\{q_k\}_{k=1}^n$, $\{g_k\}_{k=1}^{n-1}$, $\{b_k\}_{k=2}^{n-1}$, $\{h_k\}_{k=2}^n$ *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 $\widehat{x}$ is the exact solution of*

$$(A + \Delta A)\widehat{x} = b + \Delta b,$$

[3]We introduce indexes in the computed versions of the auxiliary vectors τ to clarify the analysis.

where

$$\|\Delta A(:,j)\|_2 \le \gamma_{K_1 n^2} \|A(:,j)\|_2, \quad j = 1, \dots, n, \qquad \|\Delta b\|_2 \le \gamma_{K_2 n} \|b\|_2,$$

and K_1 and K_2 are small integer constants.

Proof. Let $\widehat{z}$ 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 $\widehat{x}$ is the exact solution of a system of linear equations

$$R_{\widehat{g}+\delta g} \widehat{x} = \widehat{z}, \tag{3.22}$$

where the matrix $R_{\widehat{g}+\delta g}$ is defined as in (3.18), with the difference that in (3.19) appear the generators $\{\widehat{\bar{g}}_k\}_{k=0}^{n-1}$, $\{\widehat{\bar{b}}_k\}_{k=1}^{n-1}$, $\{\widehat{\bar{h}}_k\}_{k=1}^{n}$ computed in step 1 of `Algorithm A` instead of $\{\bar{g}_k\}_{k=0}^{n-1}$, $\{\bar{b}_k\}_{k=1}^{n-1}$, $\{\bar{h}_k\}_{k=1}^{n}$. Observe that part (b) of Theorem 3.2 can be applied with p = 5 to $R_{\widehat{g}+\delta g}$ in (3.22) and we get

$$Q^T (A + \Delta A) \widehat{x} = \widehat{z}, \quad \text{where } \|\Delta A(:,j)\|_2 \le \gamma_{K_1 n^2} \|A(:,j)\|_2,\ j = 1, \dots, n, \tag{3.23}$$

where Q is the exact orthogonal matrix defined in Theorem 3.2. It only remains to analyze the errors in the computation of $\widehat{z}$. Recall that, according to (3.6), Givens rotations we compute in `Algorithm 2` satisfy

$$\widehat{G}_i = \begin{bmatrix} \widehat{c}_i & \widehat{s}_i \\ -\widehat{s}_i & \widehat{c}_i \end{bmatrix} = \begin{bmatrix} c_i'(1+\Theta_5) & s_i'(1+\Theta_6) \\ -s_i'(1+\Theta_6) & c_i'(1+\Theta_5) \end{bmatrix},$$

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

$$\widehat{z} = (\mathcal{W}_1 \mathcal{W}_2 \cdots \mathcal{W}_{n-1})(b + \Delta b) = Q^T (b + \Delta b), \qquad \|\Delta b\|_2 \le \gamma_{K_2 n} \|b\|_2.$$

Combining this equation with (3.23) gives $Q^T(A + \Delta A)\widehat{x} = Q^T(b + \Delta b)$, and so $(A + \Delta A)\widehat{x} = b + \Delta b$. This proves the result. □

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 4.1 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]:

(4.1)

$$A = A_L + A_U = \begin{bmatrix} c_1v_1 & 0 & 0 & \cdots & 0 & 0 \\ c_2s_1v_1 & c_2v_2 & 0 & & \vdots & \vdots \\ c_3s_2s_1v_1 & c_3s_2v_2 & c_3v_3 & \ddots & & \\ \vdots & & \ddots & \ddots & 0 & 0 \\ c_{n-1}s_{n-2}\dots s_1v_1 & \cdots & & & c_{n-1}v_{n-1} & 0 \\ s_{n-1}s_{n-2}\dots s_1v_1 & \cdots & & & s_{n-1}v_{n-1} & v_n \end{bmatrix} +$$

$$+ \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & \cdots & r_{n-2}t_{n-3}\dots t_1e_1 & t_{n-2}\dots t_1e_1 \\ 0 & 0 & r_2e_2 & \cdots & r_{n-2}t_{n-3}\dots t_2e_2 & t_{n-2}\dots t_2e_2 \\ 0 & 0 & \ddots & \ddots & & \vdots \\ & & & \ddots & r_{n-2}e_{n-2} & t_{n-2}e_{n-2} \\ \vdots & \vdots & & & 0 & e_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & 0 \end{bmatrix}.$$

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

(4.2)
$$\mathbf{G} = \begin{bmatrix} c_1 & c_2 & \cdots & c_{n-1} \\ s_1 & s_2 & \cdots & s_{n-1} \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}^T,$$
$$\mathbf{H} = \begin{bmatrix} r_1 & r_2 & \cdots & r_{n-2} \\ t_1 & t_2 & \cdots & t_{n-2} \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} e_1 & e_2 & \cdots & e_{n-1} \end{bmatrix}^T.$$

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: $Q^T = W_1W_2\cdots W_{n-1}$, 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 4×4 example for simplicity. We assume as in [14] that there are no divisions by zero.

The original 4×4 matrix A is

$$A = A_L + A_U = \begin{bmatrix} c_1v_1 & 0 & 0 & 0 \\ c_2s_1v_1 & c_2v_2 & 0 & 0 \\ c_3s_2s_1v_1 & c_3s_2v_2 & c_3v_3 & 0 \\ s_3s_2s_1v_1 & s_3s_2v_2 & s_3v_3 & v_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & t_2t_1e_1 \\ 0 & 0 & r_2e_2 & t_2e_2 \\ 0 & 0 & 0 & e_3 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

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:

$$W_3A = W_3A_L + W_3A_U = \tag{4.3}$$

$$= \begin{bmatrix} c_1v_1 & 0 & 0 & 0 \\ c_2s_1v_1 & c_2v_2 & 0 & 0 \\ s_2s_1v_1 & s_2v_2 & v_3 & s_3\widetilde{v}_4 \\ 0 & 0 & 0 & c_3\widetilde{v}_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & t_2t_1e_1 \\ 0 & 0 & r_2e_2 & t_2e_2 \\ 0 & 0 & 0 & c_3e_3 \\ 0 & 0 & 0 & \widetilde{e}_4 \end{bmatrix},$$

where we introduced new parameters: $\widetilde{v}_4 = v_4$ and $\widetilde{e}_4 = -s_3e_3$, whose absolute values are the norms of the last column and the last row in W_3A_L and W_3A_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_3A_L satisfies this "wish" but matrix W_3A_U does not, because the following 2×2 block in its upper triangular part:

$$\begin{bmatrix} r_2e_2 & t_2e_2 \\ 0 & c_3e_3 \end{bmatrix}$$

has rank two in general. Fortunately, we can modify the zero entry in this 2×2 block to change the rank to one. Let us note that this zero entry in matrix W_3A_U is located at exactly the same position as the entry v_3 in matrix W_3A_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_2x_3 to 0 in the entry $(3, 3)$ of W_3A_U and subtract it from v_3 in W_3A_L. The choice of x_3 that makes the last two columns of the upper triangular part of W_3A_U semiseparable of order one is $x_3 = c_3e_3/t_2$.

If we also introduce a new parameter $\widetilde{v}_3 = v_3 - r_2x_3$ then equation (4.3) becomes:

$$W_3A = A_L^{(3)} + A_U^{(3)} = \tag{4.4}$$

$$= \begin{bmatrix} c_1v_1 & 0 & 0 & 0 \\ c_2s_1v_1 & c_2v_2 & 0 & 0 \\ s_2s_1v_1 & s_2v_2 & \widetilde{v}_3 & s_3\widetilde{v}_4 \\ 0 & 0 & 0 & c_3\widetilde{v}_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & t_2t_1e_1 \\ 0 & 0 & r_2e_2 & t_2e_2 \\ 0 & 0 & r_2x_3 & t_2x_3 \\ 0 & 0 & 0 & \widetilde{e}_4 \end{bmatrix}.$$

Applying the next Givens transformation we get

$$W_2W_3A = W_2A_L^{(3)} + W_2A_U^{(3)} =$$

$$= \begin{bmatrix} c_1v_1 & 0 & 0 & 0 \\ s_1v_1 & v_2 & s_2\widetilde{v}_3 & s_2s_3\widetilde{v}_4 \\ 0 & 0 & c_2\widetilde{v}_3 & c_2s_3\widetilde{v}_4 \\ 0 & 0 & 0 & c_3\widetilde{v}_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & t_2t_1e_1 \\ 0 & 0 & r_2(c_2e_2 + s_2x_3) & t_2(c_2e_2 + s_2x_3) \\ 0 & 0 & r_2(c_2x_3 - s_2e_2) & t_2(c_2x_3 - s_2e_2) \\ 0 & 0 & 0 & \widetilde{e}_4 \end{bmatrix},$$

where we again need to change the highlighted zero entry to some number r_1x_2 to produce semiseparability in the upper triangular part of $W_2A_U^{(3)}$:

$$W_2W_3A = A_L^{(2)} + A_U^{(2)} = \tag{4.5}$$

$$= \begin{bmatrix} c_1v_1 & 0 & 0 & 0 \\ s_1v_1 & \widetilde{v}_2 & s_2\widetilde{v}_3 & s_2s_3\widetilde{v}_4 \\ 0 & 0 & c_2\widetilde{v}_3 & c_2s_3\widetilde{v}_4 \\ 0 & 0 & 0 & c_3\widetilde{v}_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1e_1 & r_2t_1e_1 & t_2t_1e_1 \\ 0 & r_1x_2 & r_2t_1x_2 & t_1t_2x_2 \\ 0 & 0 & r_2\widetilde{e}_3 & t_2\widetilde{e}_3 \\ 0 & 0 & 0 & \widetilde{e}_4 \end{bmatrix},$$

with

$$\widetilde{e}_3 = c_2x_3 - s_2e_2, \quad x_2 = \frac{1}{t_1}(c_2e_2 + s_2x_3), \quad \widetilde{v}_2 = v_2 - r_1x_2.$$

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.

$$W_1W_2W_3A = W_1A_L^{(2)} + W_1A_U^{(2)} = \tag{4.6}$$
$$= \begin{bmatrix} v_1 & s_1\widetilde{v}_2 & s_1s_2\widetilde{v}_3 & s_1s_2s_3\widetilde{v}_4 \\ 0 & c_1\widetilde{v}_2 & c_1s_2\widetilde{v}_3 & c_1s_2s_3\widetilde{v}_4 \\ 0 & 0 & c_2\widetilde{v}_3 & c_2s_3\widetilde{v}_4 \\ 0 & 0 & 0 & c_3\widetilde{v}_4 \end{bmatrix} + \begin{bmatrix} 0 & r_1\widetilde{e}_1 & r_2t_1\widetilde{e}_1 & t_2t_1\widetilde{e}_1 \\ 0 & r_1\widetilde{e}_2 & r_2t_1\widetilde{e}_2 & t_2t_1\widetilde{e}_2 \\ 0 & 0 & r_2\widetilde{e}_3 & t_2\widetilde{e}_3 \\ 0 & 0 & 0 & \widetilde{e}_4 \end{bmatrix},$$

where $\widetilde{e}_1 = c_1e_1 + s_1x_2$ and $\widetilde{e}_2 = c_1x_2 - s_1e_1$.

All together we have the QR decomposition of matrix A expressed as:

$$A = QR, \text{ where } Q = W_3^TW_2^TW_1^T \text{ and } R = R_L + R_U,$$

where $R_L := W_1A_L^{(2)}$, $R_U := W_1A_U^{(2)}$ are given in (4.6) 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

$$R_L = \begin{bmatrix} \widetilde{v}_1 & s_1\widetilde{v}_2 & s_1s_2\widetilde{v}_3 & \cdots & s_1s_2\ldots s_{n-2}\widetilde{v}_{n-1} & s_1s_2\ldots s_{n-1}\widetilde{v}_n \\ 0 & c_1\widetilde{v}_2 & c_1s_2\widetilde{v}_3 & \cdots & c_1s_2\ldots s_{n-2}\widetilde{v}_{n-1} & c_1s_2\ldots s_{n-1}\widetilde{v}_n \\ 0 & 0 & c_2\widetilde{v}_3 & \ldots & c_2s_3\ldots s_{n-2}\widetilde{v}_{n-1} & c_2s_3\ldots s_{n-1}\widetilde{v}_n \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & c_{n-2}\widetilde{v}_{n-1} & c_{n-2}s_{n-1}\widetilde{v}_n \\ 0 & 0 & \cdots & \cdots & 0 & c_{n-1}\widetilde{v}_n \end{bmatrix} \tag{4.7}$$

and

$$R_U = \begin{bmatrix} 0 & r_1\widetilde{e}_1 & r_2t_1\widetilde{e}_1 & \cdots & r_{n-2}t_{n-3}\ldots t_1\widetilde{e}_1 & t_{n-2}\ldots t_1\widetilde{e}_1 \\ 0 & r_1\widetilde{e}_2 & r_2t_1\widetilde{e}_2 & \cdots & r_{n-2}t_{n-3}\ldots t_1\widetilde{e}_2 & t_{n-2}\ldots t_1\widetilde{e}_2 \\ 0 & 0 & r_2\widetilde{e}_3 & \ldots & r_{n-2}t_{n-3}\ldots t_2\widetilde{e}_3 & t_{n-2}\ldots t_2\widetilde{e}_3 \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \cdots & r_{n-2}\widetilde{e}_{n-1} & t_{n-2}\widetilde{e}_{n-1} \\ 0 & 0 & \cdots & \cdots & 0 & \widetilde{e}_n \end{bmatrix}. \tag{4.8}$$

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 $(R_L + R_U)x = Q^Tb$, 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.

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

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 $\{e_k\}_{k=1}^{n-1}$ of $A$ in (4.1)
  $x_n = 0$
  $\widetilde{v}_n = v_n$
  for $k = n-1$ to 2 do
    $\begin{bmatrix} x_k \\ \widetilde{e}_{k+1} \end{bmatrix} = \begin{bmatrix} \frac{1}{t_{k-1}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} e_k \\ x_{k+1} \end{bmatrix}$
    $\widetilde{v}_k = v_k - r_{k-1} x_k$
  end for
  $\begin{bmatrix} \widetilde{e}_1 \\ \widetilde{e}_2 \end{bmatrix} = \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \begin{bmatrix} e_1 \\ x_2 \end{bmatrix}$
  $\widetilde{v}_1 = v_1$
Output: parameters $\{\widetilde{v}_k\}_{k=1}^{n}, \{\widetilde{e}_k\}_{k=1}^{n}$ of $R_L$ and $R_U$ in (4.7)–(4.8).
```

Algorithm 5 Computing generators of $R = R_L + R_U$ from generators of R_L and R_U.

```
Input: generators $\{c_k\}_{k=1}^{n-1}, \{s_k\}_{k=1}^{n-1}, \{\widetilde{v}_k\}_{k=1}^{n}$ of $R_L$ in (4.7) and generators
  $\{r_k\}_{k=1}^{n-2}$, $\{t_k\}_{k=1}^{n-2}$, $\{\widetilde{e}_k\}_{k=1}^{n}$ of $R_U$ in (4.8)
  $d_1 = \widetilde{v}_1, \quad g_1 = \begin{bmatrix} s_1 & \widetilde{e}_1 \end{bmatrix}$
  for $k = 2$ to $n-1$ do
    $d_k = c_{k-1}\widetilde{v}_k + r_{k-1}\widetilde{e}_k, \quad g_k = \begin{bmatrix} c_{k-1}s_k & t_{k-1}\widetilde{e}_k \end{bmatrix}$
    $b_k = \begin{bmatrix} s_k & 0 \\ 0 & t_{k-1} \end{bmatrix}, \quad h_k = \begin{bmatrix} \widetilde{v}_k \\ r_{k-1} \end{bmatrix}$
  end for
  $d_n = c_{n-1}\widetilde{v}_n + \widetilde{e}_n, \quad h_n = \begin{bmatrix} \widetilde{v}_n \\ 1 \end{bmatrix}$
Output: generators $\{d_k\}_{k=1}^{n}, \{g_k\}_{k=1}^{n-1}, \{b_k\}_{k=2}^{n-1}, \{h_k\}_{k=2}^{n}$ of $R$ as in (2.3).
```

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.

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) $\{\widetilde{v}_1, \dots, \widetilde{v}_n\}$ of columns of R_L and norms (up to signs) $\{\widetilde{e}_1, \dots, \widetilde{e}_n\}$ 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

```
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 $\{\widetilde{v}_k\}_{k=1}^{n}, \{\widetilde{e}_k\}_{k=1}^{n}$ of $R_L$ and
          $R_U$ in (4.7)–(4.8)
  Step 2: $z = (W_1 \dots (W_{n-2}(W_{n-1}b)) \dots)$, 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 $R = R_L + R_U$ repre-
          sented as in (2.3)
  Step 4: Apply Algorithm 1 to solve $Rx = z$ and compute $x$
Output: $x_1, x_2, \dots, x_n$
```

of A in exact arithmetic in terms of R_L and R_U is

$$A = Q(R_L + R_U). \tag{4.9}$$

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 $\{\widehat{\widetilde{v}}_1, \dots, \widehat{\widetilde{v}}_n\}$ and $\{\widehat{\widetilde{e}}_1, \dots, \widehat{\widetilde{e}}_n\}$. Therefore, if $\widehat{R}_L$ and $\widehat{R}_U$ are the matrices obtained in exact arithmetic when one replaces $\widetilde{v}_i$ by $\widehat{\widetilde{v}}_i$ and $\widetilde{e}_i$ by $\widehat{\widetilde{e}}_i$, for $i = 1, \dots, n$, in (4.7) and (4.8) respectively, then the parameters computed by Algorithm 4 correspond to the QR factorization of

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

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

$$A - \widehat{B} = Q(R_L - \widehat{R}_L + R_U - \widehat{R}_U). \tag{4.10}$$

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 19.10], so let us bound the Euclidean norms of the columns of the backward error in (4.10):

$$\begin{aligned}\frac{\|A(:,i) - \widehat{B}(:,i)\|_2}{\|A(:,i)\|_2} \le & \frac{\|R_L(:,i) - \widehat{R}_L(:,i)\|_2}{\|R_L(:,i)\|_2} \frac{\|R_L(:,i)\|_2}{\|A(:,i)\|_2} \\ & + \frac{\|R_U(:,i) - \widehat{R}_U(:,i)\|_2}{\|R_U(:,i)\|_2} \frac{\|R_U(:,i)\|_2}{\|A(:,i)\|_2}, \quad i = 1, \dots, n.\end{aligned} \tag{4.11}$$

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

$$\frac{\|R_L(:,i) - \widehat{R}_L(:,i)\|_2}{\|R_L(:,i)\|_2} \approx \mathbf{u} \qquad \text{and} \qquad \frac{\|R_U(:,i) - \widehat{R}_U(:,i)\|_2}{\|R_U(:,i)\|_2} \approx \mathbf{u},$$

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

$$\max\left\{ \frac{\|R_L(:,i)\|_2}{\|A(:,i)\|_2}, \frac{\|R_U(:,i)\|_2}{\|A(:,i)\|_2} \right\} \gg 1, \qquad \text{for some } i. \tag{4.12}$$

[4]In fact, such ideal bounds do not hold for Algorithm 4.

In addition, if (4.12) holds for some index j such that $\|A(:,j)\|_2 \approx \|A\|_2$, then the whole backward error $\|A - \widehat{B}\|_2/\|A\|_2$ 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 $\{\widetilde{v}_k\}_{k=1}^{n}$, $\{\widetilde{e}_k\}_{k=1}^{n}$. Let us introduce the block-diagonal transformation matrices

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

where for $k = 1$, we define $t_0 = 1$. Then, by the virtue of `Algorithm 4`:

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

Hence, if we define vectors $\widetilde{\mathbf{e}} = \begin{bmatrix} \widetilde{e}_1 & \cdots & \widetilde{e}_n \end{bmatrix}^T$ and $\mathbf{e} = \begin{bmatrix} e_1 & \cdots & e_{n-1} & 0 \end{bmatrix}^T$, then

$$\|\widetilde{\mathbf{e}}\|_2 \le \prod_{k=1}^{n-1} \|\Theta_k\|_2 \cdot \|\mathbf{e}\|_2 = \frac{\|\mathbf{e}\|_2}{|t_1 t_2 \dots t_{n-2}|}.$$

Recall that the numbers t_k are sines of certain angles, so $|1/t_k| > 1$ and it may happen that $\|\widetilde{\mathbf{e}}\|_2 \gg \|\mathbf{e}\|_2$. To simplify the discussion, let all t_k be of magnitude t, then in the worst case $\|\widetilde{\mathbf{e}}\|_2$ would grow exponentially with n: $\|\widetilde{\mathbf{e}}\|_2 \approx \left(\frac{1}{t}\right)^{n-2} \|\mathbf{e}\|_2$. This is $\|\widetilde{\mathbf{e}}\|_2 \approx 2^{n-2}\|\mathbf{e}\|_2$, for the moderate value $t = 1/2$. In this case, observe that $\|R_U(:,i)\|_2 \ge |r_{i-1}||\widetilde{e}_i| = \frac{\sqrt{3}}{2}|\widetilde{e}_i|$ for $i = 2, \dots, n-1$. Therefore for $|\widetilde{e}_k| = \max_i |\widetilde{e}_i|$, assuming $2 \le k \le n-1$, we have for the worst case of growth of $\|\widetilde{\mathbf{e}}\|_2$ that

$$\begin{aligned} \|R_U(:,k)\|_2 &\ge \frac{\sqrt{3}}{2\sqrt{n}}\|\widetilde{\mathbf{e}}\|_2 \approx \frac{\sqrt{3}}{\sqrt{n}}2^{n-3}\|\mathbf{e}\|_2 = \\ &= \frac{\sqrt{3}}{\sqrt{n}}2^{n-3}\|A_U\|_F \approx \frac{\sqrt{3}}{\sqrt{n}}2^{n-3}\|A\|_F \ge \frac{\sqrt{3}}{\sqrt{n}}2^{n-3}\|A(:,k)\|_2, \end{aligned} \tag{4.14}$$

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 $\widehat{x}$ with large relative backward errors for $Ax = b$. As usual, we employ the relative residual $\frac{\|A\widehat{x}-b\|_2}{\|A\|_2\|\widehat{x}\|_2}$

to compute the minimum normwise relative backward error, since it is known [11, Chapter 7] that

$$\frac{\|A\widehat{x}-b\|_2}{\|A\|_2\|\widehat{x}\|_2} = \min\left\{\varepsilon : (A+\Delta A)\widehat{x} = b,\ \ \|\Delta A\|_2 \le \varepsilon\|A\|_2\right\}. \tag{4.15}$$

Traditional unstructured solvers, like Gaussian elimination with partial pivoting or QR, give $\frac{\|A\widehat{x}-b\|_2}{\|A\|_2\|\widehat{x}\|_2} \approx \mathbf{u} \approx 10^{-16}$, 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:

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

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

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 cond$(A) = 16.8185$. The right-hand side b of the system is chosen to be $b = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$. Then `Algorithm B` produces the relative residual

$$\eta = \frac{\|A\widehat{x}-b\|_2}{\|A\|_2\,\|\widehat{x}\|_2} = 1.2644\cdot 10^{-11}, \tag{4.16}$$

which is much larger than $\mathbf{u}$. 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 (4.7) and (4.8), and $M = 5.5575\cdot 10^5$, with

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

i.e., the maximum of the magnitudes in (4.12). Observe that $\eta \approx M\cdot\mathbf{u}$. 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 η and the magnitude M are plotted versus n in Figure 1. We observe again residuals much larger than $\mathbf{u}$ such that $\eta \approx M\cdot\mathbf{u}$. Note that both M and the relative residual increase exponentially with n, in agreement with (4.14).

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 100×100 order one quasiseparable matrices parameterized as in (4.1). The generators of these tests were chosen as follows: $\mathbf{G}$ and $\mathbf{H}$ in (4.2) consisted of sines and cosines of angles distributed uniformly on $[0, 2\pi]$, entries of $\mathbf{v}$ and $\mathbf{e}$ were chosen to be uniformly distributed on $[-1,1]$, as well as the entries of right-hand

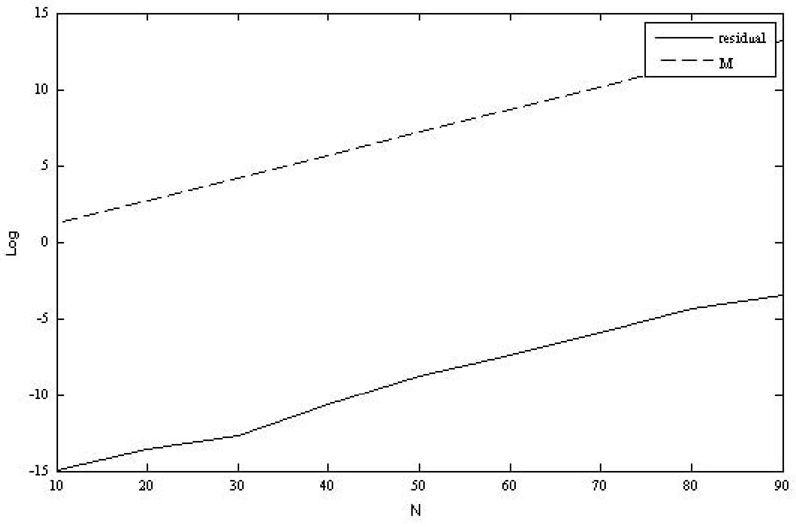

FIGURE 1. Relative residual and M versus size of matrices that have all parameters $t_k = 1/2$.

side vectors b. The relative residuals of these random tests are summarized by the histogram in Figure 2. We also show in Table 1 the maximum and minimum of the magnitude M for the cases in each bar of the histogram. Table 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.

η	e+0	e-1	e-2	e-3	e-4	e-5
M_{min}	-	7.8e+15	-	1.2e+15	1.0e+13	1.3e+11
M_{max}	-	2.6e+18	-	6.0e+15	3.2e+13	1.4e+14

η	e-6	e-7	e-8	e-9	e-10	e-11
M_{min}	2.3e+11	6.3e+9	5.8e+8	5.1e+7	3.2e+6	7.1e+5
M_{max}	9.2e+13	4.2e+13	4.3e+12	3.6e+12	1.2e+11	2.4e+11

η	e-12	e-13	e-14	e-15	e-16
M_{min}	5.1e+4	4.1e+3	4.4e+2	1.0e+2	1.3e+1
M_{max}	4.7e+9	1.8e+9	1.3e+8	2.8e+9	5.2e+6

TABLE 1. Maximum and minimum of magnitudes M for each bar of the histogram in Figure 2.

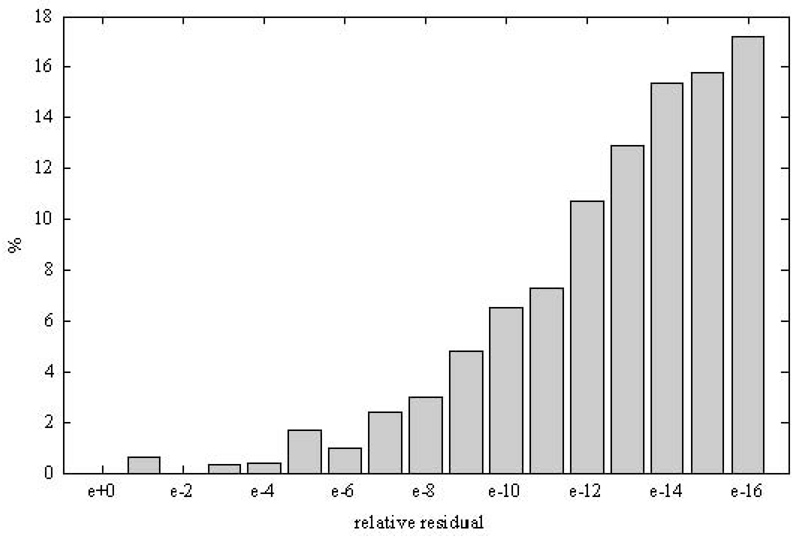

FIGURE 2. Relative residual distribution in 1000 numerical experiments with 100×100 randomly generated matrices.

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

1. T. Bella, Y. Eidelman, I. Gohberg, V. Olshevsky, and P. Zhlobich, *Classifications of recurrence relations via subclasses of (H,k)-quasiseparable matrices*, Numerical Linear Algebra in Signals, Systems and Control, Lecture Notes in Electrical Engineering, Springer-Verlag (2010).
2. S. Chandrasekaran and M. Gu, *Fast and stable algorithms for banded plus semiseparable systems of linear equations*, SIAM J. Matrix Anal. Appl. **25** (2003), no. 2, 373–384. MR 2047424 (2005f:65039)
3. S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl. **28** (2006), no. 3, 603–622. MR 2262971 (2007k:65049)
4. S. Delvaux and M. Van Barel, *A QR-based solver for rank structured matrices*, SIAM J. Matrix Anal. Appl. **30** (2008), no. 2, 464–490.
5. P. Dewilde and A.-J. van der Veen, *Time-varying systems and computations*, Kluwer Academic Publishers, Boston, MA, 1998. MR 1641480 (99g:93001)
6. Y. Eidelman and I. Gohberg, *Linear complexity inversion algorithms for a class of structured matrices*, Integral Equations and Operator Theory **35** (1999), no. 1, 28–52. MR 1707929 (2000k:65053)
7. ———, *On a new class of structured matrices*, Integral Equations and Operator Theory **34** (1999), no. 3, 293–324. MR 1689391 (2000e:15020)
8. ———, *A modification of the Dewilde-van der Veen method for inversion of finite structured matrices*, Linear Algebra Appl. **343/344** (2002), 419–450, Special issue on structured and infinite systems of linear equations. MR 1878953 (2003j:65021)
9. ———, *On generators of quasiseparable finite block matrices*, Calcolo **42** (2005), no. 3-4, 187–214. MR 2191197 (2006j:15036)
10. Y. Eidelman, I. Gohberg, and V. Olshevsky, *Eigenstructure of order-one-quasiseparable matrices. Three-term and two-term recurrence relations*, Linear Algebra Appl. **405** (2005), 1–40. MR 2148158 (2007h:15014)
11. N. J. Higham, *Accuracy and stability of numerical algorithms*, second ed., Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002. MR 1927606 (2003g:65064)
12. E. Van Camp, N. Mastronardi, and M. Van Barel, *Two fast algorithms for solving diagonal-plus-semiseparable linear systems*, J. Comput. Appl. Math. **164/165** (2004), 731–747. MR 2056911 (2005a:65029)
13. R. Vandebril, M. Van Barel, and N. Mastronardi, *A note on the representation and definition of semiseparable matrices*, Numer. Linear Algebra Appl. **12** (2005), no. 8, 839–858. MR 2172681 (2006j:15072)
14. ———, *Matrix computations and semiseparable matrices. Vol. I: Linear systems*, Johns Hopkins University Press, Baltimore, MD, 2008. MR 2378139 (2009d:15002)
15. ———, *Matrix computations and semiseparable matrices. Vol. II: Eigenvalue and singular value methods*, Johns Hopkins University Press, Baltimore, MD, 2008. MR 2460593 (2009j:15001)

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, CT 06269, USA
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`