A more accurate algorithm for computing the Christoffel transformation ⋆

María I. Bueno and Froilán M. Dopico [a,b]

[a] *Department of Mathematics, The College of William and Mary P.O. Box 8795, Williamsburg, VA 23187-8795, USA. mbueno@math.uc3m.es*

[b] *Departamento de Matemáticas, Universidad Carlos III de Madrid, Avda. de la Universidad, 30. 28911 Leganés, Spain. dopico@math.uc3m.es*

Abstract

A monic Jacobi matrix is a tridiagonal matrix which contains the parameters of the three-term recurrence relation satisfied by the sequence of monic polynomials orthogonal with respect to a measure. The basic Christoffel transformation with shift α transforms the monic Jacobi matrix associated with a measure $d\mu$ into the monic Jacobi matrix associated with $(x-\alpha)d\mu$. This transformation is known for its numerous applications to quantum mechanics, integrable systems, and other areas of mathematics and mathematical physics. From a numerical point of view, the Christoffel transformation is essentially computed by performing one step of the LR algorithm with shift, but this algorithm is not stable. We propose a more accurate algorithm, estimate its forward errors, and prove that it is forward stable, i.e., that the obtained forward errors are of similar magnitude to those produced by a backward stable algorithm. This means that the magnitude of the errors is the best one can expect, because it reflects the sensitivity of the problem to perturbations in the input data.

Key words: Christoffel transformation, forward stability, roundoff error analysis, LR algorithm.
PACS: 65G50, 65F30, 65F35, 42C05

⋆ This research has been partially supported by the Ministerio de Educación y Ciencia of Spain through grants BFM2003-06335-C03-02 (M. I. Bueno) and BFM 2003-00223(F. M. Dopico), and by the PRICIT Program of the Comunidad de Madrid through SIMUMAT Project (grant S-0505/ESP/0158). The first author has also received financial support from the Postdoctoral Fellowship EX2004-0658 provided by Ministerio de Educación y Ciencia of Spain.

1 Introduction

Let $d\mu$ be a real measure with finite moments. We say that a sequence of polynomials $\{P_n\}_{n=0}^{\infty}$ is orthogonal with respect to $d\mu$ [5] if

(1) $degree(P_n) = n$ for all $n \geq 0$.
(2) $\int_R P_n P_m d\mu = K_n \delta_{n,m}$, where $K_n \neq 0$.

In particular, $\{P_n\}_{n=0}^{\infty}$ is said to be a monic sequence of orthogonal polynomials if the coefficient of the term with higher degree of each polynomial is one. Every sequence of monic orthogonal polynomials satisfies a three-term recurrence relation

$$xP_n(x) = P_{n+1}(x) + B_{n+1}P_n(x) + G_n P_{n-1}(x),$$
$$P_{-1}(x) \equiv 0, \quad P_0(x) \equiv 1, \quad G_n \neq 0 \quad \text{for all } n.$$

The previous set of equations can be written in matrix notation in the following way

$$xp = Jp,$$

where $p = [P_0(x)\ P_1(x)\ P_2(x)\ \ldots]^T$ and

$$J = \begin{bmatrix} B_1 & 1 & 0 & \ldots \\ G_1 & B_2 & 1 & \ldots \\ 0 & G_2 & B_3 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

The semi-infinite tridiagonal matrix J is called the monic Jacobi matrix associated with $\{P_n\}$. Although it is very unusual to denote the entries of a matrix by capital letters, since we will deal with an algorithm involving two monic Jacobi matrices, for the sake of clarity, we denote by capital letters the entries in the input matrix and we denote by the same lowercase letters the entries in the output matrix.

If the measure $d\mu$ is positive the parameters G_i are positive, and the sequence of orthonormal polynomials can be considered instead of the monic one. In such a case, the corresponding Jacobi matrix is symmetric. Both Jacobi matrices, the monic and the symmetric, are related through a diagonal similarity. To keep the paper as concise as possible, we only consider monic Jacobi matrices because they cover the more general case of signed measures. Notice that in this case the Jacobi matrices may have complex eigenvalues. Parallel results hold for symmetric Jacobi matrices.

In the literature, numerous results studying the connection between the recurrence relations of polynomials orthogonal with respect to two allied measures

can be found [1,3,8,14]. This relationship can be extended to the corresponding Jacobi matrices. If $d\mu(x)$ denotes a measure and $p(x)$ denotes a polynomial, the measure given by $p(x)d\mu(x)$ is called a polynomial perturbation of $d\mu(x)$. The transformation that gives the monic Jacobi matrix associated with $p(x)d\mu(x)$ in terms of the monic Jacobi matrix associated with $d\mu(x)$ is called Christoffel or Darboux transformation. This transformation was first studied by Christoffel in 1858 [6] and by Geronimus in 1940 [12,13]. In the last two decades, this transformation has attracted the interest of various specialists in different branches of mathematics and mathematical physics for its applications to discrete integrable systems [21,26], quantum mechanics, bispectral transformation in orthogonal polynomials [16–18], and, in the context of Numerical Analysis, to the computation of quadrature rules [14,20].

Consider a monic Jacobi matrix J associated with a real measure $d\mu$ and let α be a real number. If $J - \alpha I = LU$ denotes the LU factorization without pivoting of $J - \alpha I$, where L is unit lower triangular, then the matrix version of the basic Christoffel or Darboux transformation with shift α [8,10,11,14,20] is given by

$$J - \alpha I = LU, \quad \tilde{J} = UL + \alpha I, \tag{1}$$

where $\tilde{J}$ is the monic Jacobi matrix associated with the measure $(x-\alpha)d\mu(x)$, and the factors L and U have the following structure

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots \\ l_1 & 1 & 0 & 0 & \dots \\ 0 & l_2 & 1 & 0 & \dots \\ 0 & 0 & l_3 & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots & \end{bmatrix}, \quad U = \begin{bmatrix} u_1 & 1 & 0 & 0 & \dots \\ 0 & u_2 & 1 & 0 & \dots \\ 0 & 0 & u_3 & 1 & \dots \\ 0 & 0 & 0 & u_4 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

From now on, we refer to the transformation given in (1) as Christoffel transformation with shift α.

Those readers familiar with Numerical Linear Algebra algorithms will recognize equation (1) as *one step* of the famous Rutishauser's LR algorithm to compute eigenvalues of matrices [24,25], whenever they disregard that the matrices appearing in (1) are semi-infinite. A finite version of (1) appears in equation (2) of Section 2, and it involves the leading principal submatrices of order n and $n-1$ of J and $\tilde{J}$, respectively. The transformation in equation (2) is the one we analyze from a numerical point of view. At a first glance, the relationship of (1-2) with one step of the LR algorithm discourages us of finding a numerical stable algorithm for the Christoffel transformation, because the

LR algorithm is unstable. In fact, it is well-known that the LR algorithm has been replaced by the powerful and stable QR algorithm in most eigenvalue computations [15]. One of the reasons why the LR algorithm is not stable, is that the LU factorization in (1-2) is computed without pivoting. This cannot be avoided in our case, because any pivoting strategy would destroy the tridiagonal structure needed in the Christoffel transformation. Recently, a close counterpart of the LR algorithm, the *qd* algorithm [23], was cleverly stabilized in the case of positive definite tridiagonal matrices [7]. However, the ideas in [7] cannot be directly applied to the Christoffel transformation, because in the *qd* algorithm it is assumed that the factors L and U are known. These ideas, though, give some hope to find a stable numerical algorithm for computing the Christoffel transformation. A key point in this search is that the Christoffel transformation corresponds to *only one step* of LR, and, therefore, it is not needed to stabilize the whole LR algorithm. Notice, however, that there exist some relevant differences between the Christoffel transformation and the LR algorithm: 1) the matrices $J(B, G)$ and $J(b, g)$ appearing in (2) are not similar and therefore, do not have the same eigenvalues, while the LR algorithm preserves the eigenvalues in all the steps; 2) the objective of the LR algorithm is computing the eigenvalues of the initial matrix, while the objective of the Christoffel transformation is computing the coefficients of the three-term recurrence relation corresponding to the modified sequence of orthogonal polynomials.

A direct application of (1) leads to the standard algorithm to compute Christoffel transformation. This algorithm is described in Section 2. Some authors guessed the numerical stability properties of this algorithm to compute $\tilde{J}$ based on some numerical experiments [8–10,14,20] without doing a formal error analysis. They suggested that the algorithm should be "quite stable" when the initial measure is positive and the shift does not belong to the support of the measure, i.e., all the eigenvalues of $J - \alpha I$ are real and have the same sign. A formal analysis of the stability of this algorithm was developed in [2] when $\alpha = 0$. There, it was proven that the algorithm is forward stable in this particular case. We say that an algorithm is *forward stable* if the forward errors are of similar magnitude to those produced by a backward stable algorithm [19]. However, we will show that this is not the case when $\alpha \neq 0$. The algorithm to compute Christoffel transformation based on a direct application of (1-2) is unstable, and the good results obtained in [8], [10] or [14] in the numerical experiments can be explained by the fact that the shifts used for classical families of orthogonal polynomials were close to the support of the measure associated with the Jacobi matrices.

In this work, we propose a new algorithm for computing the Christoffel transformation with shift and prove that it is more accurate than the previous one. We also estimate its forward errors with $O(n)$ cost, and prove that it is componentwise forward stable. No need to say that forward stability does not imply

small forward errors, however we will prove that the new algorithm produces componentwise relative errors of order the machine precision for large enough shifts. The forward stability result for the new algorithm holds for any type of measure –positive or signed–, and for any value of the shift for which the transformation exists. Notice that the eigenvalues of the initial Jacobi matrix may be negative or even non real numbers. It should be remarked that the new algorithm is not componentwise backward stable, even when considering positive measures supported in $(0, \infty)$, i.e, Jacobi matrices with real positive eigenvalues.

The paper is organized as follows: In Section 2 the new algorithm is introduced, and it is compared with a direct application of (1-2) on some numerical tests. These numerical experiments show that the new algorithm is more accurate. A backward error analysis of the new algorithm is presented in Section 3. A tight first order forward error bound is computed in Section 4, where it is justified why the new algorithm is more accurate than the direct application of (1-2). Finally, forward stability issues are discussed in Section 5. We warn the reader that many of the roundoff error analysis results we present require long and involved algebraic manipulations. To keep the paper concise we have only developed some of them.

2 A new algorithm for computing the Christoffel transformation

In this section we present the standard algorithm to compute the Christoffel transformation (Algorithm 1). We also present a new algorithm (Algorithm 2) for the same transformation. Moreover, we include some numerical experiments that show that the new algorithm is more accurate than the previous one.

From now on all the results refer to leading principal submatrices of monic Jacobi matrices. Since we are interested in the numerical analysis of algorithms that implement Christoffel transformation, we can only consider finite matrices. We denote by $J(B, G)$ the $n \times n$ leading principal submatrix of J, where $B = [B_1, ..., B_n]^T$, $G = [G_1, ..., G_{n-1}]^T$. Then, the finite version of the transformation given in (1) is

$$J(B, G) - \alpha I = LU, \quad J(b, g) = (UL + \alpha I)_{n-1}, \tag{2}$$

where $(M)_{n-1}$ denotes the leading principal submatrix of order $n-1$ of any matrix M, and $J(b, g)$ is the $n-1$ leading principal submatrix of $\tilde{J}$, being $b = [b_1, ..., b_{n-1}]^T$ the elements on the main diagonal of $J(b, g)$, and $g = [g_1, ..., g_{n-2}]^T$ the elements on the first lower subdiagonal, i.e., the entries in the positions $(i+1, i)$, $1 \leq i \leq n-2$. The $n \times n$ matrix equation

$J(B,G) - \alpha I = LU$ denotes the LU factorization of $J(B,G) - \alpha I$, and, here, L and U are the $n \times n$ leading principal submatrices of the semi-infinite L and U matrices appearing in (1). We have not changed the notation for L and U for the sake of simplicity.

The following pseudocode gives the standard algorithm to compute Christoffel transform with shift α of an $n \times n$ monic Jacobi matrix $J(B,G)$. This algorithm was explicitly presented by Gautschi in [10], but it also appears in matrix notation in [8] among other references.

Algorithm 1 *Given an $n \times n$ monic Jacobi matrix $J(B,G)$, this algorithm computes its Christoffel transform $J(b,g)$ of order $n-1$ with shift α.*
$u_1 = B_1 - \alpha$
for $i = 1 : n-2$
$l_i = G_i / u_i$
$b_i = u_i + l_i + \alpha$
$u_{i+1} = B_{i+1} - \alpha - l_i$
$g_i = u_{i+1} l_i$
end
$l_{n-1} = G_{n-1} / u_{n-1}$
$b_{n-1} = u_{n-1} + l_{n-1} + \alpha$

The computational cost of Algorithm 1 is $6n - 8$ flops.

Notice that Algorithm 1 is not the qd algorithm [23], [22, Section 4] since the inputs and the outputs in both algorithms are different. In the qd algorithm it is assumed that the factors L and U corresponding to the LU factorization of the original matrix are known. Therefore, the input data are, precisely, the nontrivial entries of these two matrices. Then, the qd algorithm compute the LU factorization of the matrix $J(b,g) - \alpha I$ being the factors L_1 and U_1 the output data.

Algorithm 1 may produce inaccurate outputs. Some numerical experiments will show, for instance, that when the shift α becomes larger in absolute value, the accuracy in the outputs decreases. Next we propose a slight modification in Algorithm 1 that produces a surprising improvement in the accuracy. Let us define new variables $\{t_i\}_{i=1}^{n-1}$ as $t_i := u_i + \alpha$. Then, the following new algorithm to compute Christoffel transformation with shift can be derived from Algorithm 1. Notice that the variables $u_1, ..., u_{n-1}$ have disappeared since they have been replaced by $t_1, ..., t_{n-1}$.

Algorithm 2 *Given a $n \times n$ monic Jacobi matrix $J(B,G)$, this algorithm computes its Christoffel transform $J(b,g)$ of order $n-1$ with shift α.*
$t_1 = B_1$
for $i = 1 : n-2$

$l_i = G_i/(t_i - \alpha)$
$b_i = t_i + l_i$
$t_{i+1} = B_{i+1} - l_i$
$g_i = (t_{i+1} - \alpha)\, l_i$
end
$l_{n-1} = G_{n-1}/(t_{n-1} - \alpha)$
$b_{n-1} = t_{n-1} + l_{n-1}$

The computational cost of Algorithm 2 is $6n - 9$ flops. Notice that the cost of Algorithm 2 is not larger than the cost of Algorithm 1. Moreover, the same number of divisions are performed in both algorithms.

We will show that the modifications introduced in Algorithm 1 to get Algorithm 2 have an essential influence on stability and accuracy issues. Although, we will develop a rigorous roundoff error and stability analysis of Algorithm 2 in the next sections, let us explain very briefly one of the main reasons why the new algorithm has a much better numerical behavior than the standard one. One should observe that some harmful cancellations in the computation of the outputs b_i by Algorithm 1 may arise. A significative situation where this problem can be clearly understood appears when the shift α is large: it can be easily shown that $\lim_{|\alpha|\to\infty} l_k = 0$ –see Lemma 9 in Section 4.2–, therefore $u_i = B_i - \alpha - l_{i-1} \sim -\alpha$ when $|\alpha| \to \infty$, and then $b_i = u_i + l_i + \alpha \sim (-\alpha) + \alpha$ when $|\alpha| \to \infty$. The reader should notice that this cancellation is avoided in Algorithm 2.

In matrix notation, Algorithm 2 is equivalent to

$$J(B, G) - \alpha I = L\,(T - \alpha I), \quad J(b, g) = ((T - \alpha I)\, L + \alpha I)_{n-1}\,,$$

where L is an $n \times n$ unit lower bidiagonal matrix with $l_1, ..., l_{n-1}$ in the positions $(2,1), (3,2), ..., (n, n-1)$, and T is an $n \times n$ upper bidiagonal matrix with $t_1, ..., t_n$ on the main diagonal and 1's in the positions $(1,2), (2,3), ..., (n-1, n)$. Notice that, in practice, t_n is not computed because it is not used in the computation of b and g.

Next we include some numerical experiments to show that Algorithm 2 is more accurate than Algorithm 1. We compare the forward errors produced by both algorithms for different Jacobi matrices and for different values of α. We have tested the following types of monic Jacobi matrices:

(1) A 3×3 monic Jacobi matrix with $B = [10^{-6},\ -3 \cdot 10^{-6},\ -1]$ and $G = [2 \cdot 10^{-6},\ 10^{-6}]$.
(2) Monic Jacobi matrices of dimension 30×30 associated with Laguerre polynomials with parameter $a = -19/10 + k$, where $k = 1 : 20$.
(3) Monic Jacobi matrices of dimension 30×30 associated with Jacobi poly-

nomials with parameters $a = -19/10 + k$, $b = (-9 + k)/10$, where $k = 1 : 20$.

(4) Monic Jacobi matrices of dimension 30×30 associated with Bessel polynomials with parameter $a = -101/7 + k^2$, where $k = 1 : 20$.

(5) Monic Jacobi matrix of dimension 30×30 associated with Hermite polynomials.

For each of these matrices we have computed the following *componentwise forward error*:

$$\max \left\{ \max_{k=1\ldots n-1} \left\{ \left| \frac{b_k - \hat{b}_k}{b_k} \right| \right\}, \quad \max_{k=1\ldots n-2} \left\{ \left| \frac{g_k - \hat{g}_k}{g_k} \right| \right\} \right\}, \tag{3}$$

where $\hat{b}_k$ and $\hat{g}_k$ denote the outputs computed by Algorithm 1 or 2 in standard double precision, i.e., $\epsilon \approx 1.11 \times 10^{-16}$ is the unit roundoff of the finite arithmetic, while b_k and g_k denote the outputs obtained by running the algorithms with 64 decimal digits of precision. The experiments have been done using MATLAB 5.3, and we have used the variable precision arithmetic of the Symbolic Math Toolbox of MATLAB. In all our tests, theoretical error bounds guarantee that the outputs obtained by running Algorithm 1 and 2 with 64 decimal digits of precision have more than 50 significant decimal digits.

For the different types of Jacobi matrices considered, $vm1$ and $vm2$ are vectors whose components are the componentwise forward errors obtained for each matrix by applying Algorithm 1 and 2, respectively. The results we have got are presented in Table 1, where for the sake of brevity only $\max(vm1)$ and $\max(vm2)$ are shown. Notice that the examples relative to the 3×3 matrix and the Hermite polynomials only consider one matrix for each value of α and, therefore $vm1$ and $vm2$ are just numbers. However, in Table 1 we keep the notation $\max(vm1)$ and $\max(vm2)$ for simplicity.

Notice that in most of the examples presented in Table 1 (Bessel polynomials are an exception), and for every selected value of the shift, while the forward errors from Algorithm 1 increase as the absolute value of α increases, the forward errors from Algorithm 2 decrease as $|\alpha|$ grows. Although for most of the classical families of orthogonal polynomials we have only shown results for non positive values of the shift α, the same kind of results are obtained when positive shifts are considered.

The numerical experiments we have performed indicate that the new algorithm is more accurate than Algorithm 1. In the next sections we will deduce a tight first order bound for the forward errors produced by Algorithm 2, and we will show that this bound is always smaller than a corresponding bound for Algorithm 1. The bound for Algorithm 2 is given in terms of the condition number of the problem taking into account the proper backward errors. We

3x3 Matrix	$\alpha = 1$	$\alpha = 0.3$	$\alpha = 0$	$\alpha = -1$
$\max(vm1)$	$1.3 \cdot 10^{-10}$	$1.5 \cdot 10^{-10}$	$2.2 \cdot 10^{-16}$	10^{-11}
$\max(vm2)$	$2.1 \cdot 10^{-16}$	$1.6 \cdot 10^{-15}$	$2.2 \cdot 10^{-16}$	$1.4 \cdot 10^{-16}$
Laguerre	$\alpha = 0$	$\alpha = -100$	$\alpha = -10^4$	$\alpha = -10^6$
$\max(vm1)$	$3.4 \cdot 10^{-16}$	$4.7 \cdot 10^{-14}$	$6.6 \cdot 10^{-13}$	$2.2 \cdot 10^{-10}$
$\max(vm2)$	$3.4 \cdot 10^{-16}$	$4.3 \cdot 10^{-16}$	$3.7 \cdot 10^{-16}$	$3.1 \cdot 10^{-16}$
Jacobi	$\alpha = 0$	$\alpha = -10$	$\alpha = -100$	$\alpha = -10^4$
$\max(vm1)$	$7 \cdot 10^{-13}$	$3.4 \cdot 10^{-11}$	$2.4 \cdot 10^{-10}$	$1.5 \cdot 10^{-8}$
$\max(vm2)$	$7 \cdot 10^{-13}$	$6 \cdot 10^{-14}$	$4.2 \cdot 10^{-15}$	$3 \cdot 10^{-16}$
Bessel	$\alpha = 0$	$\alpha = -10$	$\alpha = -100$	$\alpha = -10^3$
$\max(vm1)$	$3.1 \cdot 10^{-2}$	$1.2 \cdot 10^{-12}$	$1.2 \cdot 10^{-11}$	$1.3 \cdot 10^{-10}$
$\max(vm2)$	$3.1 \cdot 10^{-2}$	$1.2 \cdot 10^{-15}$	$4.3 \cdot 10^{-16}$	$4.2 \cdot 10^{-16}$
Hermite	$\alpha = 10^6$	$\alpha = 10$	$\alpha = -10^{-4}$	$\alpha = -100$
$\max(vm1)$	$2.3 \cdot 10^{-4}$	$2.6 \cdot 10^{-14}$	$7.5 \cdot 10^{-16}$	$2.7 \cdot 10^{-12}$
$\max(vm2)$	$2.2 \cdot 10^{-15}$	$3.9 \cdot 10^{-15}$	$7.5 \cdot 10^{-16}$	$6.2 \cdot 10^{-15}$

Table 1 Errors of Algorithms 1 and 2.

will also prove that when $|\alpha|$ is large enough Algorithm 2 becomes stable and accurate. Finally, we will show that the errors produced by Algorithm 2 are the best one can expect, because they reflect the sensitivity of Christoffel transformation to componentwise relative perturbations of $O(\epsilon)$ in the data.

3 Backward error analysis of Algorithm 2

We use the standard model of floating point arithmetic [19]:

$$fl(x\ op\ y) = (x\ op\ y)(1+\delta) = \frac{x\ op\ y}{1+\eta}, \quad |\delta|, |\eta| \leq \epsilon,$$

where x and y are floating point numbers, $op = +, -, *, /$, and ϵ is the unit roundoff of the machine. From now on, given a vector v, $|v|$ denotes the vector whose entries are the absolute values of the entries of v.

We develop our error analysis in the most general setting. For this purpose, we assume that the shift α is a real number, and we denote by $\hat{\alpha}$ the nearest floating point number to α. Moreover, we assume that the in-

put parameters $B_1, ..., B_{n-1}$ and $G_1, ..., G_{n-1}$ are, respectively, affected by small relative errors $(1+\epsilon_{B_1}), ..., (1+\epsilon_{B_{n-1}}), (1+\epsilon_{G_1}), ..., (1+\epsilon_{G_{n-1}})$, where $\max_{1\le i\le n-1}\{|\epsilon_{B_i}|, |\epsilon_{G_i}|\} \le \frac{C\epsilon}{1-C\epsilon}$, being C a moderate constant. These errors in the inputs may come from the rounding process when storing them in the computer. In the case of the Jacobi matrices associated with families of classical orthogonal polynomials, the inputs are computed using well-known formulae which may cause additional errors.

Theorem 1 *Let $J(B,G)$ be a monic Jacobi matrix of order n, α be a real number, and $\hat{\alpha}$ be the nearest floating point number to α. Let $J(b,g)$ be the Christoffel transform of order $n-1$ with shift α of $J(B,G)$. Let us apply Algorithm 2 to the matrix with floating point entries $J(\hat{B},\hat{G})$ where*

$$\hat{B}_i = B_i(1+\epsilon_{B_i}), \quad \hat{G}_i = G_i(1+\epsilon_{G_i}), \quad 1 \le i \le n-1,$$

and

$$\max_{1\le i\le n-1}\{|\epsilon_{B_i}|, |\epsilon_{G_i}|\} \le \frac{C\epsilon}{1-C\epsilon},$$

for a positive integer number C such that $C\epsilon \ll 1$. If $J(\hat{b},\hat{g})$ is the matrix computed by Algorithm 2, and $\hat{L}, \hat{T}$ are the computed intermediate matrices appearing in Algorithm 2, then

$$\begin{aligned} J(B+\Delta B, G+\Delta G) - \hat{\alpha}I &= \hat{L}(\hat{T}-\hat{\alpha}I), \\ J(\hat{b}+\Delta\hat{b}, \hat{g}+\Delta\hat{g}) &= \left((\hat{T}-\hat{\alpha}I)\hat{L}+\hat{\alpha}I\right)_{n-1}, \end{aligned}$$

where

$$\begin{aligned} &|\hat{\alpha}-\alpha| \le \epsilon|\alpha|, \\ &|\Delta B_i| \le \frac{(C+1)\epsilon}{1-C\epsilon}(|B_i| + |\hat{l}_{i-1}|), \quad 1 \le i \le n-1, \\ &|\Delta G_i| \le \frac{(C+2)\epsilon+\epsilon^2}{1-C\epsilon}|G_i|, \quad 1 \le i \le n-1, \\ &|\Delta\hat{b}| \le \epsilon|\hat{b}|, \quad |\Delta\hat{g}| \le (2\epsilon+\epsilon^2)|\hat{g}|. \end{aligned}$$

PROOF. For the computed quantities in the first step,

$$\hat{l}_i = \frac{G_i(1+\epsilon_{G_i})(1+\delta_{l_i})(1+\epsilon_{l_i})}{\hat{t}_i - \hat{\alpha}}, \quad |\delta_{l_i}|, |\epsilon_{l_i}| \le \epsilon.$$

Therefore,

$$|\Delta G_i| = |G_i - \hat{l}_i(\hat{t}_i - \hat{\alpha})| \le \left(\frac{(C+2)\epsilon+\epsilon^2}{1-C\epsilon}\right)|G_i|.$$

On the other hand,

$$\hat{t}_{i+1} = [B_{i+1}(1+\epsilon_{B_{i+1}}) - \hat{l}_i](1+\epsilon_{t_{i+1}}), \quad |\epsilon_{t_{i+1}}| \leq \epsilon,$$

that is,

$$|\Delta B_{i+1}| \leq \frac{(C+1)\epsilon}{1-C\epsilon}(|B_{i+1}| + |\hat{l}_i|).$$

Finally,

$$\hat{b}_i(1+\epsilon_{b_i}) = \hat{t}_i + \hat{l}_i, \quad |\epsilon_{b_i}| \leq \epsilon.$$
$$\hat{g}_i(1+\epsilon_{g_i})(1+\delta_{g_i}) = (\hat{t}_{i+1} - \hat{\alpha})\hat{l}_i, \quad |\epsilon_{g_i}|, |\delta_{g_i}| \leq \epsilon,$$

and the results follow in a straightforward way. □

In plain words Theorem 1 says that the computed Christoffel transform $J(\hat{b}, \hat{g})$ with shift α is almost the exact Christoffel transform of $J(B + \Delta B, G + \Delta G)$ with shift $\hat{\alpha}$. However the following problem arises: $|\Delta B_i|/|B_i|$ can be much larger than ϵ if $|\hat{l}_{i-1}|$ is much larger than $|B_i|$. We conclude that Algorithm 2 is componentwise stable in a mixed forward-backward sense [19] if $|\hat{l}_i| = O(|B_{i+1}|)$, for $1 \leq i \leq n-2$. Unfortunately, we cannot assure that this is the case as the following numerical experiment shows. Consider the sequence of Jacobi polynomials with parameters 1, 1/10, and the shift $\alpha = -1$. Taking into account Theorem 1, we compute a bound for the backward error as $(\epsilon \cdot errback)$, where $errback = \max_{i=2:n-1} \left\{1 + \left|\frac{\hat{l}_{i-1}}{B_i}\right|\right\}$, and we get

	$n = 10$	$n = 100$	$n = 1000$
$errback$	$1.92 \cdot 10^2$	$2 \cdot 10^4$	$2 \cdot 10^6$

The previous table shows that the upper bound of the backward error we have got is not always small. Therefore, we cannot assure mixed forward-backward stability.

4 Forward errors in Algorithm 2

The main goal of this section is to develop a bound that allows us to estimate the forward errors of Algorithm 2 in $O(n)$ operations. We will also present a result that shows that the bound for the forward errors of Algorithm 2 is always smaller than the bound for Algorithm 1. Taking this into account as well as the numerical tests in Table 1, we deduce that Algorithm 2 is more accurate than Algorithm 1. Besides, we will prove that Algorithm 2 is stable and accurate for large shifts.

To bound the errors in Algorithm 2, we study the sensitivity of Christoffel transformation with shift with respect to perturbations of the initial data, i.e., the parameters of the monic Jacobi matrix $J(B,G)$, and the shift α. We consider perturbations associated with the backward errors found in Theorem 1 and we measure the sensitivity of the problem by using the notion of componentwise relative condition number. This condition number, together with Theorem 1, allows us to get a tight upper bound on the forward errors obtained by the application of Algorithm 2 to a Jacobi matrix. This bound is presented in Theorem 3.

In the following definition the variables $l_1, l_2, \ldots, l_{n-1}$ correspond to the subdiagonal entries of the L factor in the LU factorization of $J(B,G) - \alpha I$. It should be remembered that $l_0 := 0$.

Definition 2 *Let $J(b,g)$ be the Christoffel transform of order $(n-1)$ with shift α of the $n \times n$ monic Jacobi matrix $J(B,G)$. Let $J(b+\Delta b, g+\Delta g)$ be the Christoffel transform of order $(n-1)$ with shift $\alpha+\Delta\alpha$ of the $n \times n$ monic Jacobi matrix $J(B+\Delta B, G+\Delta G)$. Let us define*

$$DB := \max\left\{ \max_{1\le i\le (n-1)} \left\{ \frac{|\Delta B_i|}{|B_i|+|l_{i-1}|} \right\}, \max_{1\le i\le (n-1)} \left\{ \frac{|\Delta G_i|}{|G_i|} \right\}, \frac{|\Delta\alpha|}{|\alpha|} \right\},$$

where the quotients $\frac{|\Delta B_i|}{|B_i|+|l_{i-1}|}$, $\frac{|\Delta G_i|}{|G_i|}$, or $\frac{|\Delta\alpha|}{|\alpha|}$ have to be understood as zero if the denominators are equal to zero. Then the relative componentwise condition number of Christoffel transformation with shift α with respect to perturbations associated with the backward errors in Theorem 1 is defined as

$$condB(J(B,G),\alpha) := \lim_{\delta\to 0} \sup_{0\le DB\le\delta} \frac{\max\left\{ \max_{1\le i\le (n-1)} \left\{ \frac{|\Delta b_i|}{|b_i|} \right\}, \max_{1\le i\le (n-2)} \left\{ \frac{|\Delta g_i|}{|g_i|} \right\} \right\}}{DB}.$$

The condition number $condB(J(B,G),\alpha)$ is infinite if some of the denominators appearing in the relative changes of the outputs b_i, i.e, $\frac{|\Delta b_i|}{|b_i|}$, is zero. However $b_i = 0$ will only happen for extremely particular values of the shift α. In these cases, other condition numbers have to be considered. For instance, measuring absolute changes in the corresponding components of b, or measuring relative normwise changes of b. We do not consider these particular situations in this work. Notice that $g_i \neq 0$ for all i since $g_i = (t_{i+1} - \alpha) l_i$ and both factors $t_{i+1} - \alpha$ and l_i are nonzero.

The condition number $condB(J(B,G),\alpha)$ allows us to give an upper bound on the forward errors produced by Algorithm 2, as the following theorem shows.

Theorem 3 *Let $J(b,g)$ and $J(\hat{b},\hat{g})$ be, respectively, the exact and the com-*

puted Christoffel transform with shift α of $J(B,G)$ by Algorithm 2, then

$$\max_k \left\{ \left| \frac{b_k - \hat{b}_k}{b_k} \right|, \left| \frac{g_k - \hat{g}_k}{g_k} \right| \right\} \leq (C+2)\epsilon \left(1 + condB(J(B,G),\alpha)\right) + O(\epsilon^2),$$

where the left hand side of the previous inequality is a shorthand expression for (3).

The proof of this theorem is a straightforward consequence of Theorem 1. We will provide a way to compute $condB(J(B,G),\alpha)$, and therefore a bound on the forward errors, with $O(n)$ cost. It is essential to remark that we have checked on the reliability of the bound on the forward errors running many numerical experiments, where we have observed that the bound does not overestimate significantly the actual errors.

The entries b and g of the Christoffel transform $J(b,g)$ of $J(B,G)$ are rational functions of the inputs B, G, and α, and, as a consequence, b and g are differentiable functions of B, G, and α whenever the denominators are different from zero. Therefore, $condB(J(B,G),\alpha)$ can be expressed in terms of partial derivatives [4]. More precisely:

$$condB(J(B,G),\alpha) = \max\{\max_{1\leq k\leq n-1}\{condB(b_k)\}, \max_{1\leq k\leq n-2}\{condB(g_k)\}\}, \quad (4)$$

where

$$condB(b_k) = \sum_{i=1}^{k} \left| \frac{|B_i| + |l_{i-1}|}{b_k} \frac{\partial b_k}{\partial B_i} \right| + \sum_{i=1}^{k} \left| \frac{G_i}{b_k} \frac{\partial b_k}{\partial G_i} \right| + \left| \frac{\alpha}{b_k} \frac{\partial b_k}{\partial \alpha} \right|, \quad (5)$$

$$condB(g_k) = \sum_{i=1}^{k+1} \left| \frac{|B_i| + |l_{i-1}|}{g_k} \frac{\partial g_k}{\partial B_i} \right| + \sum_{i=1}^{k} \left| \frac{G_i}{g_k} \frac{\partial g_k}{\partial G_i} \right| + \left| \frac{\alpha}{g_k} \frac{\partial g_k}{\partial \alpha} \right|. \quad (6)$$

In Theorem 8, we will get recurrence relations for $condB(b_k)$ and $condB(g_k)$ that lead to an explicit expression for $condB(J(B,G),\alpha)$. Our first step to prove Theorem 8 is to express the intermediate variables l_k in Algorithm 2, and the outputs b_k and g_k as functions of the data B, G and α. Then, we obtain expressions for the partial derivatives of each of these functions with respect to their arguments.

From Algorithm 2, we get

$$l_k = \frac{G_k}{B_k - \alpha - l_{k-1}} \quad (7)$$

and, therefore, l_k can be seen as a function of $B_1, ..., B_k, G_1, ..., G_k, \alpha$.

Lemma 4 *If α is a real number such that $J(B,G)-\alpha I$ has a unique LU factorization, then l_k has the following partial derivatives with respect to $B_1,...,B_k$, $G_1,...,G_k$ and α*

$$\frac{\partial l_k}{\partial B_i} = \begin{cases} \dfrac{l_k}{t_k-\alpha}\dfrac{\partial l_{k-1}}{\partial B_i}, & i<k, \\ -\dfrac{l_k}{t_k-\alpha}, & i=k. \end{cases}$$

$$\frac{\partial l_k}{\partial G_i} = \begin{cases} \dfrac{l_k}{t_k-\alpha}\dfrac{\partial l_{k-1}}{\partial G_i}, & i<k, \\ \dfrac{1}{t_k-\alpha}, & i=k. \end{cases}$$

$$\frac{\partial l_k}{\partial \alpha} = \begin{cases} \dfrac{l_k}{t_k-\alpha}\left(1+\dfrac{\partial l_{k-1}}{\partial \alpha}\right), & 1<k, \\ \dfrac{l_1}{t_1-\alpha}, & k=1. \end{cases}$$

PROOF. For $i<k$,

$$\frac{\partial l_k}{\partial B_i} = \frac{G_k}{(B_k-\alpha-l_{k-1})^2}\frac{\partial l_{k-1}}{\partial B_i} = \frac{l_k(t_k-\alpha)}{(t_k-\alpha)^2}\frac{\partial l_{k-1}}{\partial B_i}.$$

For $i=k$,

$$\frac{\partial l_k}{\partial B_k} = \frac{-G_k}{(B_k-\alpha-l_{k-1})^2}.$$

The rest of the formulas are obtained in a similar way. □

From Algorithm 2, we also get

$$b_k = B_k + l_k - l_{k-1} \tag{8}$$

and, therefore, b_k can be seen as a function of $B_1,...,B_k,G_1,...,G_k,\alpha$. It also happens that

$$g_k = (B_{k+1}-\alpha-l_k)l_k. \tag{9}$$

Notice that g_k is a function of $B_1,...,B_{k+1},G_1,...,G_k,\alpha$.

Lemma 5 *If α is a real number such that $J(B,G)-\alpha I$ has a unique LU factorization, then the partial derivatives of b_k with respect to $B_1,...,B_k$, $G_1,...,G_k$, and α are*

$$\frac{\partial b_k}{\partial B_i} = \begin{cases} \left(\dfrac{l_k}{t_k-\alpha}-1\right)\dfrac{\partial l_{k-1}}{\partial B_i}, & i<k, \\ 1-\dfrac{l_k}{t_k-\alpha}, & i=k. \end{cases}$$

$$\frac{\partial b_k}{\partial G_i} = \begin{cases} \left(\dfrac{l_k}{t_k-\alpha}-1\right)\dfrac{\partial l_{k-1}}{\partial G_i}, & i<k, \\ \dfrac{1}{t_k-\alpha}, & i=k. \end{cases}$$

$$\frac{\partial b_k}{\partial \alpha} = \begin{cases} \dfrac{l_k}{t_k-\alpha}+\left(\dfrac{l_k}{t_k-\alpha}-1\right)\dfrac{\partial l_{k-1}}{\partial \alpha}, & 1<k, \\ \dfrac{l_1}{t_1-\alpha}, & k=1. \end{cases}$$

PROOF. From (8), for $i \leq k-1$,

$$\frac{\partial b_k}{\partial B_i} = \frac{\partial l_k}{\partial B_i} - \frac{\partial l_{k-1}}{\partial B_i}.$$

Taking into account Lemma 4, the first result follows. For $i=k$,

$$\frac{\partial b_k}{\partial B_k} = 1 + \frac{\partial l_k}{\partial B_k} = 1 - \frac{l_k}{t_k-\alpha}.$$

The results for $\frac{\partial b_k}{\partial G_i}$ and $\frac{\partial b_k}{\partial \alpha}$ are obtained in a similar way. $\square$

Lemma 6 *If α is a real number such that $J(B,G)-\alpha I$ has a unique LU factorization, then the partial derivatives of g_k with respect to $B_1,...,B_{k+1}$,*

$G_1, ..., G_k$, *and* α *are*

$$\frac{\partial g_k}{\partial B_i} = \begin{cases} (t_{k+1} - \alpha - l_k)\,\dfrac{\partial l_k}{\partial B_i}, & i \le k, \\ l_k, & i = k+1. \end{cases}$$

$$\frac{\partial g_k}{\partial G_i} = (t_{k+1} - \alpha - l_k)\,\frac{\partial l_k}{\partial G_i}, \quad i \le k.$$

$$\frac{\partial g_k}{\partial \alpha} = -l_k + (t_{k+1} - \alpha - l_k)\frac{\partial l_k}{\partial \alpha}, \quad \textit{for } k \ge 1.$$

PROOF. Taking into account (9), for $i \le k$,

$$\frac{\partial g_k}{\partial B_i} = \left(-\frac{\partial l_k}{\partial B_i}\right) l_k + (B_{k+1} - \alpha - l_k)\frac{\partial l_k}{\partial B_i},$$

and the first result follows. The results for $\frac{\partial g_k}{\partial G_i}$ and $\frac{\partial g_k}{\partial \alpha}$ are obtained in the same way. □

Next we define some quantities that will be useful to give a recursive formula for the condition number $condB(J(B,G),\alpha)$. Let us call

$$condB_{BG}(l_k) := \sum_{i=1}^{k} condB_{B_i}(l_k) + \sum_{i=1}^{k} cond_{G_i}(l_k), \tag{10}$$

where

$$condB_{B_i}(l_k) := \left|\frac{|B_i| + |l_{i-1}|}{l_k}\frac{\partial l_k}{\partial B_i}\right|, \quad \text{and} \quad cond_{G_i}(l_k) := \left|\frac{G_i}{l_k}\frac{\partial l_k}{\partial G_i}\right|. \tag{11}$$

The quantities $condB_{BG}(l_k)$ can be computed recursively as the following lemma shows.

Lemma 7 *Let* α *be a real number such that* $J(B,G) - \alpha I$ *has a unique* LU *factorization. Then, for* $k \ge 1$,

$$condB_{BG}(l_k) := 1 + \left|\frac{B_k}{t_k - \alpha}\right| + \left|\frac{l_{k-1}}{t_k - \alpha}\right| (1 + condB_{BG}(l_{k-1})),$$

where $condB_{BG}(l_0) := 0$.

PROOF. If $i < k$,

$$condB_{B_i}(l_k) = \frac{|B_i| + |l_{i-1}|}{|l_k|} \left| \frac{l_k}{t_k - \alpha} \right| \left| \frac{\partial l_{k-1}}{\partial B_i} \right| = \left| \frac{l_{k-1}}{t_k - \alpha} \right| condB_{B_i}(l_{k-1}).$$

If $i = k$,

$$condB_{B_k}(l_k) = \frac{|B_k| + |l_{k-1}|}{|l_k|} \left| \frac{l_k}{t_k - \alpha} \right| = \left| \frac{|B_k| + |l_{k-1}|}{t_k - \alpha} \right|.$$

If $i < k$,

$$cond_{G_i}(l_k) = \left| \frac{G_i}{l_k} \right| \left| \frac{l_k}{t_k - \alpha} \right| \left| \frac{\partial l_{k-1}}{\partial G_i} \right| = \left| \frac{l_{k-1}}{t_k - \alpha} \right| cond_{G_i}(l_{k-1}).$$

If $i = k$,

$$cond_{G_k}(l_k) = \left| \frac{G_k}{l_k} \right| \left| \frac{1}{t_k - \alpha} \right| = 1.$$

These expressions lead us to the recurrence relation for $condB_{BG}(l_k)$ in an straightforward way. □

Theorem 8 gives recurrence relations that, taking into account (4), allow us to compute $condB(J(B,G),\alpha)$ in $O(n)$ flops. We gather all the recurrence relations needed to perform this computation in the statement of Theorem 8.

Theorem 8 *Let $J(B,G)$ be any $n \times n$ Jacobi matrix, and let α be a real number such that $J(B,G) - \alpha I$ has a unique LU factorization. Let L be the lower bidiagonal factor in the LU factorization of $J(B,G) - \alpha I$ and $T := U + \alpha I$, where U is the upper bidiagonal factor in the same factorization. If $l_1, l_2, ..., l_{n-1}$ are the entries of L in positions $(2,1),(3,2),...,(n,n-1)$ and $t_1, t_2, ..., t_{n-1}$ are the entries of T in positions $(1,1),(2,2),...,(n-1,n-1)$, then*

$$\begin{aligned} condB(b_k) = {} & \left| \frac{l_k}{b_k} \right| + \left| \frac{l_k - t_k + \alpha}{b_k} \right| \left(\left| \frac{B_k}{t_k - \alpha} \right| + \left| \frac{l_{k-1}}{t_k - \alpha} \right| (1 + condB_{BG}(l_{k-1})) \right) \\ & + \left| \frac{\alpha}{t_k - \alpha} \right| \left| \frac{l_k}{b_k} + \left(\frac{l_k - t_k + \alpha}{b_k} \right) \frac{\partial l_{k-1}}{\partial \alpha} \right|, \\ condB(g_k) = {} & \left| \frac{l_k}{t_{k+1} - \alpha} \right| + \left| \frac{B_{k+1}}{t_{k+1} - \alpha} \right| + \left| 1 - \frac{l_k}{t_{k+1} - \alpha} \right| condB_{BG}(l_k) \\ & + \left| \frac{\alpha}{g_k} \right| \left| -l_k + (t_{k+1} - \alpha - l_k) \frac{\partial l_k}{\partial \alpha} \right|, \end{aligned}$$

where $condB_{BG}(l_0) := 0$,

$$condB_{BG}(l_k) := 1 + \frac{|B_k|}{|t_k - \alpha|} + \left|\frac{l_{k-1}}{t_k - \alpha}\right| (1 + condB_{BG}(l_{k-1})),$$

and

$$\frac{\partial l_k}{\partial \alpha} = \begin{cases} \dfrac{l_k}{t_k - \alpha}\left(1 + \dfrac{\partial l_{k-1}}{\partial \alpha}\right), & 1 < k, \\ \dfrac{l_1}{t_1 - \alpha}, & k = 1. \end{cases}$$

Notice that the variables appearing in the previous recurrence relations are the same we introduced in Algorithm 2. Assuming that these variables are known, the computational cost of computing $condB(J(B, G), \alpha)$ is $33n - 61$. In order to obtain this cost, it is important to realize that some operations appear more than once in the computation of $condB_{BG}(l_k)$, $condB(b_k)$ and $condB(g_k)$, and a careful counting of these quantities is necessary to keep the cost low.

PROOF. Let us call

$$condB_{B_i}(b_k) := \left|\frac{|B_i| + |l_{i-1}|}{b_k}\frac{\partial b_k}{\partial B_i}\right|,$$

$$cond_{G_i}(b_k) := \left|\frac{G_i}{b_k}\frac{\partial b_k}{\partial G_i}\right|,\ cond_{\alpha}(b_k) := \left|\frac{\alpha}{b_k}\frac{\partial b_k}{\partial \alpha}\right|.$$

Then, using Lemma 5 and taking into account (11), if $i < k$

$$\begin{aligned} condB_{B_i}(b_k) &= \left|\frac{|B_i| + |l_{i-1}|}{b_k}\right| \left|\frac{l_k}{t_k - \alpha} - 1\right| \left|\frac{\partial l_{k-1}}{\partial B_i}\right| \\ &= \left|\frac{l_{k-1}}{b_k}\right| \left|\frac{l_k}{t_k - \alpha} - 1\right| condB_{B_i}(l_{k-1}), \end{aligned}$$

and

$$condB_{B_k}(b_k) = \left|\frac{|B_k| + |l_{k-1}|}{b_k}\right| \left|\frac{l_k}{t_k - \alpha} - 1\right|.$$

Similarly, if $i < k$,

$$cond_{G_i}(b_k) = \left|\frac{G_i}{b_k}\right| \left|\frac{l_k}{t_k - \alpha} - 1\right| \left|\frac{\partial l_{k-1}}{\partial G_i}\right| = \left|\frac{l_{k-1}}{b_k}\right| \left|\frac{l_k}{t_k - \alpha} - 1\right| cond_{G_i}(l_{k-1}),$$

and $cond_{G_k}(b_k) = \left|\frac{l_k}{b_k}\right|$. Finally,

$$cond_\alpha(b_k) = \left|\frac{\alpha}{b_k}\right| \left| \frac{l_k}{t_k - \alpha} + \left(\frac{l_k}{t_k - \alpha} - 1\right) \frac{\partial l_{k-1}}{\partial \alpha} \right|.$$

Taking into account (5) and (10), the result follows for $condB(b_k)$. The result for $condB(g_k)$ can be obtained in a similar way. □

4.1 Comparison with error bounds for Algorithm 1

It is possible to develop a roundoff error analysis of Algorithm 1 similar to the analysis done for Algorithm 2. To begin with, backward error bounds for Algorithm 1 can be found. Then, it is also possible to deduce recurrence relations for a relative componentwise condition number, $condA(J(B,G),\alpha)$, for Christoffel transformation with respect to perturbations in the input data associated with the backward errors of Algorithm 1. Finally, the condition number $condA(J(B,G),\alpha)$ can be used in a counterpart version of Theorem 3 for Algorithm 1 to bound the forward errors. We do not include the details of these results to keep the paper concise. However, we would like to remark that it is easy to prove that

$$condB(J(B,G),\alpha) \leq condA(J(B,G),\alpha),$$

for all monic Jacobi matrices $J(B,G)$ and all shifts α. This fact, together with the numerical experiments in Section 2, show that Algorithm 2 is more accurate than Algorithm 1.

4.2 Stability and accuracy for large shifts

There are some interesting results that we can prove related to the stability and accuracy of Algorithm 2 beyond the fact of being more accurate than Algorithm 1. It can be proven that, for large enough absolute values of the shift, Algorithm 2 is *accurate*, i.e., it produces outputs with componentwise forward errors of order $O(\epsilon)$. To prove this, we will show that $condB(J(B,G),\alpha)$ tends to 3 when $|\alpha|$ tends to ∞. Therefore, Theorem 3 guarantees accuracy in this situation. The numerical experiments in Section 2 show that this is not the case for Algorithm 1. In fact, it can be proven that Algorithm 1 decreases its accuracy as $|\alpha|$ grows.

Let us remember that, according to Theorem 1, if $|\hat{l}_{i-1}| = O(|B_i|)$ for $1 \leq i \leq n-1$, then Algorithm 2 is mixed forward-backward stable, which is the usual

requirement for a numerical algorithm to be considered stable [19, p. 7]. More precisely, in this case, it can be said that the computed Christoffel transform $J(\hat{b}, \hat{g})$ with shift α of $J(B, G)$ is an $O(\epsilon)$ relative componentwise perturbation of the *exact* Christoffel transform with shift $\hat{\alpha}$ of $J(B + \Delta B, G + \Delta G)$, where ΔB and ΔG are $O(\epsilon)$ relative componentwise perturbations of the *exact* inputs B and G. In this context, another goal of this subsection is to prove that for large enough values of the shift, $|l_{i-1}| \ll |B_i|$ and then Algorithm 2 is stable. We have to admit that this will be proven for the exact values of $|l_{i-1}|$ and not for the computed values $|\hat{l}_{i-1}|$, thus we are only proving stability up to $O(\epsilon^2)$ terms.

Accuracy and stability are both based on the following simple Lemma.

Lemma 9 *Let l_k, $1 \leq k \leq n-1$, be the variables appearing in Algorithm 2, i.e., the subdiagonal elements in the L factor of the LU factorization of $J(B, G) - \alpha I$. Then*

$$\lim_{|\alpha| \to \infty} |l_k| = 0.$$

As a consequence Algorithm 2 is stable for $|\alpha|$ large enough.

PROOF. Notice that $|l_1| = \left|\frac{G_1}{B_1 - \alpha}\right|$ and $\lim_{|\alpha| \to \infty} |l_1| = 0$. Let us proceed by using induction. Assume that $\lim_{|\alpha| \to \infty} |l_{k-1}| = 0$, then

$$\lim_{|\alpha| \to \infty} |l_k| = \lim_{|\alpha| \to \infty} \left|\frac{G_k}{B_k - \alpha - l_{k-1}}\right| = 0.$$

□

Theorem 10 *Let $condB(J(B, G), \alpha)$ be the condition number for Christoffel transformation with shift introduced in Definition 2. Then*

$$\lim_{|\alpha| \to \infty} condB(J(B, G), \alpha) = 3.$$

This implies that Algorithm 2 is accurate for $|\alpha|$ large enough.

PROOF. Let us remember that $t_k = B_k - l_{k-1}$. Lemma 9 implies

$$\lim_{|\alpha| \to \infty} \left|\frac{B_k}{t_k - \alpha}\right| = 0, \quad \text{and} \quad \lim_{|\alpha| \to \infty} \left|\frac{l_{k-1}}{t_k - \alpha}\right| = 0, \quad k \geq 1.$$

Then, taking into account Lemma 7,

$$\lim_{|\alpha| \to \infty} condB_{BG}(l_k) = 1, \quad k \geq 1.$$

Consider now the following limits

$$\lim_{|\alpha|\to\infty}\left|\frac{l_k}{b_k}\right| = \lim_{|\alpha|\to\infty}\left|\frac{G_k}{(B_k-\alpha-l_{k-1})(B_k+l_k-l_{k-1})}\right| = 0,$$

$$\lim_{|\alpha|\to\infty}\left|\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{B_k}{t_k-\alpha}\right| = \lim_{|\alpha|\to\infty}\left|\frac{G_k-(t_k-\alpha)^2}{(t_k-\alpha)(B_k-l_{k-1})+G_k}\cdot\frac{B_k}{t_k-\alpha}\right| = 1,$$

$$\lim_{|\alpha|\to\infty}\left|\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{l_{k-1}}{t_k-\alpha}\right| =$$

$$= \lim_{|\alpha|\to\infty}\left|\frac{G_k-(t_k-\alpha)^2}{(t_k-\alpha)(B_k-l_{k-1})+G_k}\cdot\frac{G_{k-1}}{(t_{k-1}-\alpha)(t_k-\alpha)}\right| = 0,$$

$$\lim_{|\alpha|\to\infty}\left|\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{\partial l_{k-1}}{\partial\alpha}\right| = \lim_{|\alpha|\to\infty}\left|\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{l_{k-1}}{t_{k-1}-\alpha}\left(1+\frac{\partial l_{k-2}}{\partial\alpha}\right)\right|.$$

Applying induction and Lemma 4, it is easy to show that $\lim_{|\alpha|\to\infty}\frac{\partial l_i}{\partial\alpha}=0$, for all i. Moreover,

$$\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{l_{k-1}}{t_{k-1}-\alpha}\frac{G_k-(t_k-\alpha)^2}{(t_k-\alpha)(B_k-l_{k-1})+G_k}\cdot\frac{G_{k-1}}{(t_{k-1}-\alpha)^2}.$$

Therefore,

$$\lim_{|\alpha|\to\infty}\left|\frac{l_k-t_k+\alpha}{b_k}\cdot\frac{\partial l_{k-1}}{\partial\alpha}\right| = 0.$$

Taking into account Theorem 8 and the previous limits we see that

$$\lim_{|\alpha|\to\infty} condB(b_k) = 1, \quad k\geq 1. \tag{12}$$

Finally, we consider the following limits

$$\lim_{|\alpha|\to\infty}\left|\frac{\alpha\cdot l_k}{g_k}\right| = \lim_{|\alpha|\to\infty}\left|\frac{\alpha}{t_{k+1}-\alpha}\right| = 1,$$

$$\lim_{|\alpha|\to\infty}\left|\frac{\alpha}{t_{k+1}-\alpha}\cdot\frac{t_{k+1}-\alpha-l_k}{l_k}\cdot\frac{\partial l_k}{\partial\alpha}\right| =$$

$$\lim_{|\alpha|\to\infty}\left|\frac{\alpha}{t_{k+1}-\alpha}\cdot\frac{t_{k+1}-\alpha-l_k}{t_k-\alpha}\cdot\left(1+\frac{\partial l_{k-1}}{\partial\alpha}\right)\right| = 1.$$

Taking into account Theorem 8 and the previous limits, we get

$$\lim_{|\alpha|\to\infty} condB(g_k) = 3, \quad k\geq 1 \tag{13}$$

Therefore, considering (12), (13), and (4), the result follows. □

Similarly, it can be proven that $condA(J(B,G),\alpha)$ tends to infinity when $|\alpha|$ grows.

So far, we have shown that Algorithm 2 produces smaller forward errors than Algorithm 1 and we have proven that these forwards errors become $O(\epsilon)$ for large enough shifts.

5 Algorithm 2 is forward stable

The purpose of this section is to prove that the forward error bound we have found for Algorithm 2 is the best one can expect, because it reflects the sensitivity of Christoffel transformation to componentwise relative perturbations in the data. However, we have seen that Algorithm 2 is not backward stable, then we are forced to use a weaker notion of stability. According to [19, p. 9], an algorithm is said to be *forward stable* if it produces answers with forward errors of similar magnitude to those produced by a backward stable algorithm. In this section we show that Algorithm 2 is componentwise forward stable. In order to prove that, we define the relative componentwise condition number of Christoffel transformation with shift α with respect to small componentwise relative perturbations of B, G, and α

$$condC(J(B,G),\alpha) = \lim_{\delta\to 0} \sup_{0\le DC\le\delta} \frac{\max\left\{\max\limits_{1\le i\le(n-1)}\left\{\dfrac{|\Delta b_i|}{|b_i|}\right\}, \max\limits_{1\le i\le(n-2)}\left\{\dfrac{|\Delta g_i|}{|g_i|}\right\}\right\}}{DC}, \quad (14)$$

where

$$DC = \max\left\{\max_{1\le i\le(n-1)}\left\{\frac{|\Delta B_i|}{|B_i|}\right\}, \max_{1\le i\le(n-1)}\left\{\frac{|\Delta G_i|}{|G_i|}\right\}, \frac{|\Delta\alpha|}{|\alpha|}\right\}.$$

To prove that Algorithm 2 is componentwise forward stable is equivalent to prove that $condC(J(B,G),\alpha)$ and $condB(J(B,G),\alpha)$ have the same order of magnitude, by taking into account Theorem 3.

Recurrent expressions for $condC(J(B,G),\alpha)$ can be obtained in a similar way as we got recurrent expressions for $condB(J(B,G),\alpha)$. By using these expressions, we can prove Theorem 11, after considerably long and delicate algebraic manipulations are performed. This theorem states that the condition numbers, $condB(J(B,G),\alpha)$ and $condC(J(B,G),\alpha)$, we have considered for

the Christoffel transformation with shift are of the same order of magnitude, which implies that Algorithm 2 is forward stable.

Theorem 11 *Let $condB(J(B,G),\alpha)$ and $condC(J(B,G),\alpha)$ be the condition numbers introduced, respectively, in Definition 2 and (14) for the Christoffel transformation with shift, then*

$$condC(J(B,G),\alpha) \leq condB(J(B,G),\alpha) \leq 8\, condC(J(B,G),\alpha). \qquad (15)$$

This result together with the fact that $condB(J(B,G),\alpha) \geq 1$ implies that Algorithm 2 is componentwise forward stable.

References

[1] Bueno, M.I. and Marcellán, F. *Darboux transformation and perturbation of linear functionals*, Linear Algebra and Applications, **384** 215–242(2004).

[2] Bueno, M.I. and Dopico, F.M, *Stability and sensitivity of Darboux transformation without parameter*, Electron. Trans. Numer. Anal., **18**, 101–136(2004).

[3] Buhmann, M. and Iserles, A., *On orthogonal polynomials transformed by the QR algorithm*, Journal Comp. Appl. Math. **43**, 117–134(1992).

[4] Chaitin-Chatelin, and F., Frayssé V., *Lectures on finite precision computations*, SIAM, Philadelphia (1996).

[5] Chihara, T.S., *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[6] Christoffel,E.B., *Über die Gaubische Quadratur und eine Verallgemeinerung derselben*, J. Reine Angew. Math., **55**, 61–82(1858).

[7] Fernando, K. V., and Parlett B. N., *Accurate singular values and differential qd algorithms*, Numer. Math., **67**, 191–229(1994).

[8] Galant, D., *An implementation of Christoffel's Theorem in the Theory of Orthogonal Polynomials*, Math. Comput., **25**, 111–113(1971).

[9] Galant, D., *Algebraic Methods for modified orthogonal polynomials*, Math. Comput., **59**, 541–546(1992).

[10] Gautschi, W., *An algorithmic implementation of the generalized Christoffel theorem*, Numerical Integration (G. Hämmerlin, ed.), Internat. Ser. Numer. Math., **57**, 89–106(1982).

[11] Gautschi, W., *The interplay between classical analysis and (numerical) linear algebra- A tribute to Gene H. Golub*, Electron. Trans. Numer. Anal., **13**, 119–147(2002).

[12] Geronimus, Ya. L., *On the polynomials orthogonal with respect to a given number sequence and a theorem*, ed W. Hahn, Izv. Akad. Nauk, **4**, 215-228(1940). (in Russian)

[13] Geronimus, Ya. L., *On the polynomials orthogonal with respect to a given number sequence*, Zap. Mat. Otdel. Khar'kov. Univers. i NII Mat. i Mehan., **17**, 3-18(1940).

[14] Golub, G.H., and Kautsky, J., *Calculation of Gauss Quadratures with Multiple Free and Fixed Knots*, Numer. Math., **41**, 147–163(1983).

[15] Golub, G. H., and Van Loan, C. F., *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, 1996.

[16] Grünbaum, F.A., and Haine, L., *Orthogonal Polynomials satisfying Differential Equations: the Role of the Darboux Transformation*, CRM Proceedings and Lecture Notes, American Mathematical Society. Providence, Rhode Island. **9**, 143–154(1996).

[17] Grünbaum, F.A., and Haine, L., *Bispectral Darboux Transformations:An Extension of the Krall Polynomials*, Internat. Math. Res. Notices, **8**, 359–392(1997).

[18] Grünbaum, F.A, Haine, L., and Horozov, E, *Some functions that generalize the Krall-Laguerre polynomials*, J. of Comp. Appl. Math., **106**, 271–297 (1999).

[19] Higham, N. J., *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, 2002.

[20] Kautsky, J., and Golub, G.H., *On the calculation of Jacobi Matrices*, Linear Algebra Appl., **52/53**, 439–455(1983).

[21] Matveev, V.B., and Salle, M.A., *Differential-Difference Evolution Equations. II (Darboux Transformation for the Toda Lattice)* , Letters in Math. Physics, **3**, 425–429(1979).

[22] Parlett, B.N., *The new qd algorithms*, Acta Numerica, 459–491(1995).

[23] Rutishauser, H., *Der Quotienten-Differenzen-Algorithmus*, Z. Angew. Math. Phys., **5**, 233–251(1954).

[24] Rutishauser, H., *Solution of eigenvalue problems with the LR transformation*, Nat. Bur. Standards Appl. Math. Ser., **49**, 47–81(1958).

[25] Rutishauser, H., *Lectures on Numerical Mathematics*, Birkhäuser, Boston, 1990.

[26] Spiridonov, V., Vinet, L. and Zhedanov, A., *Spectral transformations, self-similar reductions and orthogonal polynomials*, J. Phys. A: Math. & Gen., **30**, 7621–7637(1997).