Accurate polynomial root–finding methods for symmetric tridiagonal matrix eigenproblems

L. Gemignani

Dipartimento di Informatica, Università di Pisa, Largo Bruno Pontecorvo, 3 - 56127 Pisa, Italy

Abstract

In this paper we consider the application of polynomial root-finding methods to the solution of the tridiagonal matrix eigenproblem. All considered solvers are based on evaluating the Newton correction. We show that the use of scaled three-term recurrence relations complemented with error free transformations yields some compensated schemes which significantly improve the accuracy of computed results at a modest increase in computational cost. Numerical experiments illustrate that under some restriction on the conditioning the novel iterations can approximate and/or refine the eigenvalues of a tridiagonal matrix with high relative accuracy.

Keywords: Polynomial zeros, tridiagonal matrix, matrix eigenvalues, three-term recurrence, floating-point arithmetic, accuracy.
2010 MSC: 65F15

1. Introduction

Polynomial root-finding algorithms can be applied for the solution of structured matrix eigenproblems. Most of the methods including the Newton method for eigenvalue refinement and the Ehrlich-Aberth iteration for simultaneous eigenvalue computation [3] need at each step to evaluate only the ratio $f(z)/f'(z)$ –generally referred as the Newton correction– between the value of the characteristic polynomial and of its first derivative. It is an immediate observation that the function value and the derivative might overflow/underflow while the ratio may still be a reasonable machine number. Numerically reliable polynomial methods should be able to exploit the structure of the matrix eigenproblem for the efficient and accurate evaluation of the Newton correction.

In this paper we focus on the symmetric tridiagonal eigenproblem. It is well known that in some cases the efficiency of the polynomial solver can be coupled with the high accuracy of the computed approximations. Theoretical and computational results have been already established in the literature for the

Email address: `l.gemignani@di.unipi.it` (L. Gemignani) important subclass of real symmetric tridiagonal matrices with zero diagonal entries. Such matrices arise naturally both in the framework of divide-and-conquer methods for bidiagonal singular value problems [12, 15, 19] and in the approximation theory for the computation of Gauss-type quadrature rules for symmetric weight functions [20, 13]. Since symmetric tridiagonal matrices with zero diagonal specify their eigenvalues with high relative accuracy independently of their magnitudes [6] numerical methods can possibly compute these eigenvalues at the same relative accuracy they are determined by the input data. Relatively accurate polynomial zerofinders based on Laguerre's iteration are considered in [19, 24] while GR-type matrix methods are devised in [6, 8]. Similar results do not hold for a general symmetric tridiagonal whose entries do not determine its eigenvalues to high relative accuracy. Higher (multi)precision computation can be recommended to increase the accuracy of the computed approximations with some timing penalties.

It is a classical result that the characteristic polynomial of a tridiagonal matrix together with its derivatives can efficiently be evaluated by using three-term recurrences [25]. Although such recurrences are computationally appealing and straightforward to implement in practice the resulting scheme can be prone to numerical difficulties. Due to overflow and underflow occurrences in real applications the computation needs to incorporate some normalization or scaling techniques [26]. In addition, the three-term sequence computation is backward stable but this does not imply any accuracy in the evaluation of the polynomial and its derivative and, a fortiori, of the corresponding Newton correction. According to the classical rule of thumb the (relative) forward error depends both on the (relative) backward error and the condition number. Wilkinson analyzed the three-term recurrence computation [25] by proving that the evaluation of the characteristic polynomial is relatively backward stable for points close to the origin. Nevertheless, quite commonly computing the determinant of a symmetric tridiagonal matrix is an ill-conditioned problem.

A similar situation also occurs with the Ruffini-Horner algorithm generally used to evaluate polynomials and incorporated in the multiprecision polynomial rootfinder MPSolve [2]. Though named for Paolo Ruffini (1765-1822) and William Horner (1786-1837), two European mathematicians who described it in the early 1800s, the Ruffini-Horner method had first appeared in mathematical texts from both Arab and Chinese medieval mathematicians [4] and then rediscovered in 1669 by Isaac Newton (see [21]). Very recently an accurate variant of the Ruffini-Horner scheme has been proposed in [10] which is capable to compute a result of the same quality as if computed using twice the working precision and then rounded to the working precision. This variant makes use of some modified algorithms –called error-free transformations in [22]– for evaluating the sum and the product of two floating point numbers introduced by Knuth [17] and Dekker–Veltkamp [5], respectively.

In this contribution we combine error-free transformations and scaled three-term recurrence relations to produce an efficient and accurate algorithm for evaluating both the characteristic polynomial of a symmetric tridiagonal matrix and its first derivative. By "accurate" we mean that the computed answers have relative errors as they were computed in twice the working precision. This means that we achieve full precision accuracy, apart from severely ill-conditioned computations, without timing penalties required by multiprecision environments. Then the algorithm is incorporated in the Newton method for testing purposes. By using a result in [23] the accurate computation of the characteristic polynomial implies that the same property holds for the approximation of the eigenvalues. Our numerical experience suggests that the resulting rootfinder is typically able to approximate matrix eigenvalues at high relative accuracy independently of their magnitude.

The paper is organized as follows. In Section 2 we introduce and analyze the compensated variants of the classical schemes making use of three-term recurrences for evaluating the characteristic polynomial, as well as its first derivative, of tridiagonal matrices. In Section 3 we illustrate the results of numerical experiments confirming the potential high relative accuracy of an eigenvalue refinement method based on the Newton method complemented with the compensated techniques for computing the function values. Finally, conclusion and future work are drawn in Section 4.

2. Accurate three-term recurrence computation

Let $T \in \mathbb{R}^{n\times n}$ be a symmetric unreduced tridiagonal matrix, i.e.,

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}, \quad \beta_i, \alpha_i \in \mathbb{R},\ \beta_i \neq 0,\ 1 \le i \le n-1.$$

The characteristic polynomial

$$f_n(\lambda) = \det(\lambda I_n - T)$$

can be computed by the three-term recurrence relations

$$\begin{aligned} &f_0(\lambda) = 1, \quad f_1(\lambda) = \lambda - \alpha_1; \\ &f_j(\lambda) = (\lambda - \alpha_j) f_{j-1}(\lambda) - \beta_{j-1}^2 f_{j-2}(\lambda), \quad j = 2, 3, \dots, n. \end{aligned}$$

By differentiating the relations we obtain a second recurrence for the evaluation of the first derivative $f'_n(\lambda)$, namely,

$$\begin{aligned} &f'_0(\lambda) = 0, \quad f'_1(\lambda) = 1; \\ &f'_j(\lambda) = f_{j-1}(\lambda) + (\lambda - \alpha_j) f'_{j-1}(\lambda) - \beta_{j-1}^2 f'_{j-2}(\lambda), \quad j = 2, 3, \dots, n. \end{aligned}$$

The MatLab[1] function `evalpoly1` (**Algorithm 1**) [9] computes the function values $f_n(\lambda)$ and $f'_n(\lambda)$ for a given λ and returns the value of the Newton correction given by $r = \dfrac{f_n(\lambda)}{f'_n(\lambda)}$.

[1]Matlab is a registered trademark of The MathWorks, Inc..

Algorithm 1: `Evaluating the characteristic polynomial and its derivative`

```
function r=evalpoly1($\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}$);
n=length($\boldsymbol{\alpha}$);
f1s=0; f1=1; f2s=1; f2=$\lambda - \boldsymbol{\alpha}(1)$;
for k=1:n-1
  f0s=f1s; f0=f1;
  f1s=f2s; f1=f2;
  $\beta = \boldsymbol{\beta}(k)$; $\beta = \beta$*$\beta$; $\gamma = \lambda - \boldsymbol{\alpha}(k+1)$;
  f2s=f1+ $\gamma$*f1s-$\beta$*f0s;
  f2=$\gamma$*f1-$\beta$*f0;
end
r=f2/f2s;
```

The previous **Algorithm 1** in the iterative phase only performs additions and multiplications. As has been noticed by several authors (see [5] and the references given therein) the errors generated by a floating point addition and multiplication are always floating point numbers. Specifically, let $\mathcal{F}$ be the set of floating point numbers, ϵ is the machine precision and, as usual, let $fl(\cdot)$ denote the result of a floating point computation. Then the *elementary rounding error* e generated in the computation of $fl(a \circ b)$, $a, b \in \mathcal{F}$, $\circ \in \{+, -, *\}$, namely,

$$e = (a \circ b) - fl(a \circ b)$$

satisfies $e \in \mathcal{F}$. The algorithms **TwoSum** [17] and **TwoProduct** [5] given in input two floating point numbers a and b return as output the pair of floating numbers $fl(a \circ b)$ and e for $\circ = +$ and $\circ = *$, respectively. The properties of these algorithms are summarized in the following theorem [22].

Theorem 2.1. *Let $a, b \in \mathcal{F}$ and let $x, y \in \mathcal{F}$ be such that $[x, y] =$ **TwoSum**(a, b). Then*

$$a + b = x + y, \quad x = fl(a + b), \quad |y| \le \epsilon|x|, \quad |y| \le \epsilon|a + b|.$$

The algorithm **TwoSum** *requires 6 flops.*

*Let $a, b \in \mathcal{F}$ and let $x, y \in \mathcal{F}$ be such that $[x, y] =$ **TwoProduct**(a, b). Then*

$$a * b = x + y, \quad x = fl(a * b), \quad |y| \le \epsilon|x|, \quad |y| \le \epsilon|a * b|.$$

The algorithm **TwoProduct** *requires 17 flops.*

Now, for the sake of simplicity let us assume that the input data $(\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta})$ in **Algorithm 1** are floating point numbers and the function `evalpoly1` is modified to return just only the value `f2` of the characteristic polynomial. Scaling the matrix T by a diagonal similarity transform yields

$$\widehat{T} = \begin{bmatrix} \alpha_1 & \beta_1^2 & & \\ 1 \ddots & \ddots & & \\ & \ddots & \ddots & \beta_{n-1}^2 \\ & & 1 & \alpha_n \end{bmatrix},$$

which says that `f2` is really a function of λ, α_i and β_i^2. For the sake of simplicity also suppose that $\beta_j^2 \in \mathcal{F}$ for $j = 1, 2, \ldots, n-1$. Then the function $g(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}) := $ `evalpoly1`$(\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta})$, with $\hat{\boldsymbol{\beta}} = \left[\beta_1^2, \ldots, \beta_{n-1}^2\right]$, is computed by means of an algorithm $\hat{g}$ using intermediate quantities $g_1, \ldots, g_N$ with corresponding elementary rounding errors $y_1, \ldots, y_N$ and returning g_{N+1} as output. Thus we can write $g_{N+1} = \hat{g}(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}, \boldsymbol{y})$, $\boldsymbol{y} = [y_1, \ldots, y_N]$. By a first-order analysis since $g(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}) = \hat{g}(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}, \boldsymbol{0})$ it follows that

$$\Delta = g(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}) - \hat{g}(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}, \boldsymbol{y}) = -\sum_{i=1}^{N} \frac{\partial \hat{g}}{\partial y_i}(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}, \boldsymbol{y}) y_i + O(\parallel \boldsymbol{y} \parallel_\infty^2). \quad (2.1)$$

The approach pursued by the compensated schemes proposed in [18, 10, 16] consists of computing the correcting term $\hat{y} = -\sum_{i=1}^{N} \frac{\partial \hat{g}}{\partial y_i}(\lambda, \boldsymbol{\alpha}, \hat{\boldsymbol{\beta}}, \boldsymbol{y}) y_i$ which gives the first-order approximation of Δ and then returning the corrected result $\tilde{g}$ defined by

$$\tilde{g} = fl(g_{N+1} + \hat{y}).$$

A compensated variant of **Algorithm 1** should work as described in the next **Algorithm 2**. A slightly different version can be found in [11].

The following observation motivates our analysis.

Remark 2.2. *As long as we assume that $\beta_j, \beta_j^2 \in \mathcal{F}$ for $j = 1, 2, \ldots, n-1$, then it is easily seen that the values of Γ and Δ returned as output by* **Algorithm 2** *provide first-order approximations of the global forward error generated in the computation of the characteristic polynomial and its first derivative, respectively. In addition, if we look at the zero diagonal case where $\alpha_j = 0$, $1 \leq j \leq n$, then the global forward errors are linear function of the elementary rounding errors so that the big O term in* (2.1) *is zero and the linear terms gives the exact corrections.*

To support theoretically the intuitive argument in favor of the compensation technique we present in the sequel a forward rounding error analysis of **Algorithm 2**. A forward error analysis of three-term recurrences is somehow customary. We outline below the main points by taking in mind the application of **Algorithm 2** to polynomial rootfinding.

Let $(\delta_0, \delta_2, \delta_3, \delta_4) := (\delta_0^{(k)}, \delta_2^{(k)}, \delta_3^{(k)}, \delta_4^{(k)})$, $k = 1, 2, \ldots, n-1$, be the quadruple of elementary rounding errors generated at step k of **Algorithm 2** in the

Algorithm 2: Compensated variant of **Algorithm 1**

```
function rc=evalpoly1c($\lambda, \boldsymbol{\alpha}, \boldsymbol{\beta}$);
n=length($\boldsymbol{\alpha}$);
f1s=0; f1=1; f2s=1; [f2,$\delta_0$]=TwoSum[$\lambda, -\boldsymbol{\alpha}(1)$];
$\Delta_1 = 0$; $\Delta = 0$; $\Gamma_1 = \delta_0$; $\Gamma = 0$;
for k=1:n-1
   $\Delta_0 = \Delta_1$; $\Delta_1 = \Delta$;
   $\Gamma_0 = \Gamma_1$; $\Gamma_1 = \Gamma$;
   f0s=f1s; f0=f1;
   f1s=f2s; f1=f2;
   $\beta = \boldsymbol{\beta}(k)$; $\beta = \beta * \beta$;
   [$\gamma, \delta_0$]=TwoSum[$\lambda, -\boldsymbol{\alpha}(k+1)$];
   [r0s,$\delta_2$]=TwoProduct[$\beta$,f0s]; [r1s, $\delta_3$]=TwoProduct[$\gamma$,f1s];
   [r2s, $\delta_4$]=TwoSum[f1,r1s]; [f2s, $\delta_5$]=TwoSum[r2s, -r0s];
   $\Delta = \Gamma_1 + \gamma * \Delta_1 + \delta_0 * \texttt{f1s} - \beta * \Delta_0 + \delta_3 + \delta_4 + \delta_5 - \delta_2$;
   [r0, $\delta_2$]=TwoProduct[$\beta$,f0]; [r1, $\delta_3$]=TwoProduct[$\gamma$,f1];
   [f2, $\delta_4$]=TwoSum[r1, -r0];
   $\Gamma = \gamma * \Gamma_1 + \delta_0 * \texttt{f1} - \beta * \Gamma_0 + \delta_3 + \delta_4 - \delta_2$;
end
f2=f2+$\Gamma$; f2s=f2s +$\Delta$; rc=f2/f2s;
```

computation of the value of the characteristic polynomial. Set $\hat{f}_k(\lambda) = fl(f_k(\lambda))$ and $\mu_k = \delta_0^{(k)} \hat{f}_k(\lambda) + \delta_3^{(k)} + \delta_4^{(k)} - \delta_2^{(k)}$. From Theorem 2.1 by calling

$$\Gamma_k = f_k(\lambda) - \hat{f}_k(\lambda), \quad k = 0, 1, \ldots, n,$$

we obtain

$$\begin{aligned} &\Gamma_0 = 0, \quad \Gamma_1 = \mu_0 = \delta_0^{(0)}; \\ &\Gamma_{k+1} = (\lambda - \alpha_{k+1})\Gamma_k - \beta_k^2 \Gamma_{k-1} + \mu_k, \quad k = 1, 2, \ldots, n-1. \end{aligned}$$

The computation of the sequence Γ_k can be recasted in matrix form as the solution of the following banded linear system with coefficient matrix $F \in \mathbb{R}^{(n+1)\times(n+1)}$,

$$F\boldsymbol{\Gamma} = \boldsymbol{\mu}, \quad \begin{bmatrix} 1 & \alpha_n - \lambda & \beta_{n-1}^2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_1^2 \\ & & & \ddots & \alpha_1 - \lambda \\ & & & & 1 \end{bmatrix} \begin{bmatrix} \Gamma_n \\ \Gamma_{n-1} \\ \vdots \\ \Gamma_1 \\ \Gamma_0 \end{bmatrix} = \begin{bmatrix} \mu_{n-1} \\ \vdots \\ \mu_1 \\ \mu_0 \\ 0 \end{bmatrix}. \tag{2.2}$$

The matrix characterization yields the next result.

Theorem 2.3. *Let us define the sequence of complementary orthogonal polynomials given by*

$$\begin{aligned}&\rho_0(\lambda) = 1, \quad \rho_1(\lambda) = \lambda - \alpha_n;\\ &\rho_j(\lambda) = (\lambda - \alpha_{n-j+1})\rho_{j-1}(\lambda) - \beta_{n-j+1}^2 \rho_{j-2}(\lambda), \quad j = 2, 3, \ldots, n-1.\end{aligned}$$

Then we find that

$$\Gamma_n = \sum_{j=0}^{n-1} \mu_{n-1-j}\rho_j(\lambda). \tag{2.3}$$

Furthermore, we have

$$|\Gamma_k| \leq 3\epsilon\ \Gamma^* + O(\epsilon^2), \quad \Gamma^* = M \parallel F^{-1} \parallel_\infty, \quad k = 0, 1, \ldots, n,$$

where

$$M = \max_k \{(|\lambda| + |\alpha_{k+1}|)|f_k(\lambda)| + \beta_k^2|f_{k-1}(\lambda)|\}.$$

Proof. Let us consider the persymmetric partitioning of F defined by

$$F = \left[\begin{array}{c|c} \boldsymbol{e}_1 & \hat{T}_n \\ \hline 0 & \boldsymbol{e}_n^T \end{array}\right].$$

The first characterization of Γ_n follows by using Schur complementation applied to this partitioning in order to compute a block representation of the inverse of F. Concerning the second estimate let us observe that from Theorem 2.1 we obtain that

$$|\mu_k| \leq 3\epsilon(|\lambda - \alpha_{k+1}||\hat{f}_k(\lambda)| + \beta_k^2|\hat{f}_{k-1}(\lambda)|) + O(\epsilon^2), \quad k = 1, 2, \ldots, n-1,$$

and, hence,

$$|\mu_k| \leq 3\epsilon M + O(\epsilon^2), \quad k = 1, 2, \ldots, n-1.$$

Then from (2.2) it follows that at the first order

$$|\Gamma_k| \leq 3\epsilon M \parallel F^{-1} \parallel_\infty = 3\epsilon\Gamma^*, \quad k = 0, 1, \ldots, n.$$

□

Equation (2.3) describes the connection between three-term recurrences and Clenshaw's algorithm for the summation of certain finite series. A related forward error analysis of Clenshaw's algorithm can be found in [7, 1]. The final value of `f2` returned by **Algorithm 2** satisfies

$$\mathtt{f2} = fl(\hat{f}_n(\lambda) + fl(\Gamma_n)) = (\hat{f}_n(\lambda) + fl(\Gamma_n))(1 + \epsilon_1), \quad |\epsilon_1| \leq \epsilon.$$

A forward error analysis of the algorithm for evaluating Γ_n can be derived similarly as above. By setting

$$\gamma_k = \Gamma_k - fl(\Gamma_k), \quad k = 0, 1, \ldots, n,$$

we obtain

$$\begin{aligned}&\gamma_0 = 0, \quad \gamma_1 = 0;\\ &\gamma_{k+1} = (\lambda - \alpha_{k+1})\gamma_k - \beta_k^2\gamma_{k-1} + \nu_k, \quad k = 1, 2, \ldots, n-1,\end{aligned}$$

where

$$|\nu_k| \leq 3\epsilon(|\lambda - \alpha_{k+1}||\Gamma_k| + \beta_k^2|\Gamma_{k-1}| + |\mu_k|) + O(\epsilon^3), \quad k = 1, 2, \ldots, n-1.$$

Hence, from Theorem 2.3 it is found that

$$\gamma_n = \sum_{j=0}^{n-2} \nu_{n-1-j}\rho_j(\lambda),$$

and, moreover,

$$|\nu_k| \leq 9\epsilon^2 M \parallel F^{-1} \parallel_\infty (|\lambda - \alpha_{k+1}| + \beta_k^2 + \parallel F \parallel_\infty) + O(\epsilon^3), \quad k = 1, 2, \ldots, n-1.$$

In this way, we finally arrive at the following estimate

$$\texttt{f2} = (\hat{f}_n(\lambda) + \Gamma_n - \gamma_n)(1 + \epsilon_1) = f_n(\lambda)(1 + \epsilon_1) - \gamma_n(1 + \epsilon_1), \quad |\epsilon_1| \leq \epsilon,$$

and, equivalently,

$$|\texttt{f2} - f_n(\lambda)| \leq \epsilon|f_n(\lambda)| + |\gamma_n| + O(\epsilon^3), \tag{2.4}$$

where

$$|\gamma_n| \leq 9\epsilon^2\Gamma^*(\sum_{j=0}^{n-2}(|\lambda - \alpha_{n-1-j}| + \beta_{n-1-j}^2 + \parallel F \parallel_\infty)|\rho_j(\lambda)|).$$

It is worth mentioning that the value of Γ^* determines the conditioning of the zero–finding problem for the polynomial $f_n(\lambda)$. Indeed, if we consider relative perturbations of the values $\beta_i^2 \to \beta_i^2(1 + \epsilon_i)$ and $\alpha_i \to \alpha_i(1 + \epsilon_i')$ $|\epsilon_i|, |\epsilon_i'| \leq \epsilon''$, and the corresponding perturbed characteristic polynomial $\tilde{f}_n(\lambda)$ then it can be shown that for any simple root ξ of $f_n(\lambda)$ and a sufficiently small ϵ'' there is a simple root $\tilde{\xi}$ of $\tilde{f}_n(\lambda)$ such that

$$\frac{|\tilde{\xi} - \xi|}{|\xi|} \leq 3\epsilon\frac{\Gamma^*}{|\xi||f_n'(\xi)|} + O(\epsilon^2).$$

Hence by applying corollary 2.3 in [23] we obtain the following result.

Theorem 2.4. *Assume that $f_n(\xi) = 0$, $f_n'(\xi) \neq 0$, $\xi \in \mathbb{R}$, and, moreover, there exists $I = [\xi - \delta, \xi + \delta]$, $\delta \geq 0$, such that $\forall\lambda \in I$ we have*

$$\epsilon \; \frac{|fl(f_n'(\lambda)) - f_n'(\lambda)|}{|f_n'(\lambda)|} \leq 1/8,$$

and

$$\frac{\max_{\lambda\in I}|f_n''(\lambda)|}{|f_n'(\xi)|}|\lambda-\xi| \le 1/8.$$

Let us consider the Newton method applied in floating point arithmetic for the solution of $f_n(\lambda)=0$ by making use of **Algorithm2** *for computing the function values. Suppose that no premature underflow condition occurs for initial guesses $\xi_0 \in I$. Then there exists $I' \subset I$ such that $\forall \xi_0 \in I'$ the method applied with starting point ξ_0 generates a sequence $\{\xi_k\}$ whose relative error decreases until the first k for which*

$$\frac{|\xi_k-\xi|}{|\xi|} \simeq \epsilon + 9\epsilon^2 N \frac{\Gamma^*}{|\xi||f_n'(\xi)|},$$

where

$$N = \max_{\lambda\in I}\sum_{j=0}^{n-2}(|\lambda-\alpha_{n-1-j}|+\beta_{n-1-j}^2+\parallel F\parallel_\infty)|\rho_j(\lambda)|.$$

Incorporating scaling techniques in **Algorithm2** can be useful in order to prevent underflow/overflow situations and to keep N of moderate magnitude. This leads to the following modified version –referred as to **Algorithm 3**– which will be tested experimentally in the next section.

It is remarkable to notice that the value of the Newton correction is independent of the scaling factor so that there is no need to compute back the “exact” evaluations of the characteristic polynomial and its first derivative.

3. Numerical Results

We have designed a numerical method for eigenvalue refinement of symmetric tridiagonal matrices based on the Newton iteration complemented with **Algorithm 3** for the evaluation of the function values. The resulting algorithm named **New_cs** –acronym of Newton's method with compensation and scaling– have been tested in Matlab. For comparison purposes we have also implemented a variant called **New_s** making use of **Algorithm 1** complemented with the same scaling strategy as employed in **Algorithm 3**. Numerical results are shown to illustrate the differences between the compensated algorithm and this latter variant.

To measure the relative error of a computed approximation $\widehat{\xi}$ we rely upon the following well known inclusion theorem for polynomial roots (see Corollary 6.4g in [14]). Assume that $\widehat{\xi} \neq 0$ and $f_n'(\widehat{\xi}) \neq 0$ then there exists a root ξ of $f_n(\lambda)$ such that

$$\frac{|\widehat{\xi}-\xi|}{|\widehat{\xi}|} \le n\frac{|f_n(\widehat{\xi})|}{|f_n'(\widehat{\xi})||\widehat{\xi}|}.$$

The relative approximation error `app_err` of a computed nonzero approximation $\widehat{\xi}$ is therefore defined as

$$\texttt{app\_err}:\ = n\frac{|f_n(\widehat{\xi})|}{|f_n'(\widehat{\xi})||\widehat{\xi}|}.$$

Algorithm 3: Compensated variant of **Algorithm 1** with scaling

```
function rcs=evalpoly1c($\lambda, \boldsymbol{\beta}$, Ms, Mi);
n=length($\boldsymbol{\beta}$); n=n+1;
f1s=0; f1=1; f2s=1; [f2,$\delta_0$]=TwoSum[$\lambda, -\boldsymbol{\alpha}(1)$];
$\Delta_1 = 0$; $\Delta = 0$; $\Gamma_1 = \delta_0$; $\Gamma = 0$;
for k=1:n-1
  $\Delta_0 = \Delta_1$; $\Delta_1 = \Delta$;
  $\Gamma_0 = \Gamma_1$; $\Gamma_1 = \Gamma$;
  f0s=f1s; f0=f1;
  f1s=f2s; f1=f2;
  $\beta = \boldsymbol{\beta}(k)$; $\beta = \beta * \beta$;
  [$\gamma, \delta_0$]=TwoSum[$\lambda, -\boldsymbol{\alpha}(k+1)$];
  [r0s,$\delta_2$]=TwoProduct[$\beta$,f0s]; [r1s, $\delta_3$]=TwoProduct[$\gamma$,f1s];
  [r2s, $\delta_4$]=TwoSum[f1,r1s]; [f2s, $\delta_5$]=TwoSum[r2s, -r0s];
  $\Delta = \Gamma_1 + \gamma * \Delta_1 + \delta_0 * \mathtt{f1s} - \beta * \Delta_0 + \delta_3 + \delta_4 + \delta_5 - \delta_2$;
  [r0, $\delta_2$]=TwoProduct[$\beta$,f0]; [r1, $\delta_3$]=TwoProduct[$\gamma$,f1];
  [f2, $\delta_4$]=TwoSum[r1, -r0];
  $\Gamma = \gamma * \Gamma_1 + \delta_0 * \mathtt{f1} - \beta * \Gamma_0 + \delta_3 + \delta_4 - \delta_2$;
  w=max{|f1|, |f2|};
  if (w $\geq$ Ms)
    w=Ms/w; w=ceil($\log_2$(w)); w=$2^{\mathtt{w}}$;
    f1=f1*w; f2=f2*w; f1s=f1s*w; f2s=f2s*w;
    $\Delta = \Delta$*w; $\Delta 1 = \Delta 1$*w; $\Gamma = \Gamma$*w; $\Gamma 1 = \Gamma 1$*w;
    else
      if (w $\leq$ Mi)
        w=Msi/w; w=floor($\log_2$(w)); w=$2^{\mathtt{w}}$;
        f1=f1*w; f2=f2*w; f1s=f1s*w; f2s=f2s*w;
        $\Delta = \Delta$*w; $\Delta 1 = \Delta 1$*w; $\Gamma = \Gamma$*w; $\Gamma 1 = \Gamma 1$*w;
      end;
  end;
end;
f2=f2+$\Gamma$; f2s=f2s +$\Delta$; rcs=f2/f2s;
```

where the Newton correction is evaluated by **Algorithm 1** carried out in high (128 digits) precision arithmetic. Both algorithms **New_cs** and **New_s** are stopped whenever the relative error between two consecutive iterates is less than ϵ in magnitude or the number of iterations exceed a given fixed `maxit` value. According to [26] the values of `Ms` and `Mi` used in **Algorithm 3** for scaling are set to 2^{34} and 2^{-34}, respectively, in our implementations.

Out test suite consists of the following examples:

1. The tridiagonal reduction of the `rosser matrix`, that is, $T = \mathtt{hess}(\mathtt{rosser})$. The `rosser` matrix is a small 8×8 classical test for matrix eigenvalue algorithms. The matrix, and thus also T, has a numerically zero eigenvalue.

2. Random symmetric tridiagonal matrices $T_n \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1 = \epsilon$, $\lambda_i = 1+(i-2)\sqrt{\epsilon}$, $2 \le i \le n$. The matrices are obtained by Householder tridiagonal reduction of random dense symmetric matrices that have the given eigenvalue distribution.

3. Tridiagonal Toeplitz matrices $T_n = (t_{i,j}) \in \mathbb{R}^{n\times n}$ with

$$t_{i,j} = \begin{cases} 1 \text{ if } |i-j| = 1; \\ 2-(\pi/(n+1))^2 \text{ if } |i-j| = 0. \end{cases}$$

   These matrices have one small eigenvalue of order $1/n^4$.

4. Shifted Wilkinson tridiagonal matrices $T_n(\alpha) \in \mathbb{R}^{n\times n}$ generated by

$$T_n(\alpha) = \texttt{wilkinson}(n) - \alpha \cdot \texttt{eye}(n).$$

5. Symmetric tridiagonal matrices $T_n \in \mathbb{R}^{n\times n}$, $T_n = T_n(\alpha,\beta)$, $n = 2k$, with zero diagonal and subdiagonal entries defined by

$$\beta_{2j-1} = \alpha \in \{2^{-10}, 1\},\ \beta_{2j} = \beta = \{1, 4n\}, \quad j = 1, 2, \ldots, k.$$

   These matrices are encountered when computing the SVD of bidiagonal Toeplitz matrices [8]. The eigenvalues occur in pairs $(-\lambda, \lambda)$ with typically two paired eigenvalues very close to the origin.

In all the tests performed we start the iterative refinement using the approximation of the eigenvalue of minimum modulus returned by the function `eig` applied to the input matrix.

In Example 1 both algorithms **New_cs** and **New_s** stop in three iterations. The approximation error is `app_err` = 4.6e−17 for the approximation returned by **New_cs** and `app_err` = 4.3e−2 for the approximation returned by **New_s**.

For the input matrices of Example 2 algorithm **New_s** does not generally fulfill the stopping condition on the relative error by thus performing `maxit` iterations in all the tests considered without any significant improvement of the accuracy. Differently algorithm **New_cs** stops in a few (less than 5) iterations. In next Figure 1 we show the approximation errors measured at the end of execution of algorithm **New_cs**.

Similar results can be found for the matrices in Example 3. In Figure 2 we plot the approximation errors measured at the end of execution of algorithm **New_cs**. Observe that the accuracy of the approximation returned by the compensated scheme is quite independent of its magnitude.

Wilkinson's eigenvalue test matrices are symmetric tridiagonal matrices with pairs of nearly equal eigenvalues and unit subdiagonal entries. We consider the following two classes of shifted matrices:

1. $T_o = \texttt{wilkinson}(21 + 20 \cdot k) - 6 \cdot \texttt{eye}(21 + 20 \cdot k)$, $k = 1, 2, \ldots$;

2. $T_e = \texttt{wilkinson}(32 + 20 \cdot k) - 8.5 \cdot \texttt{eye}(22 + 20 \cdot k)$, $k = 1, 2, \ldots$.

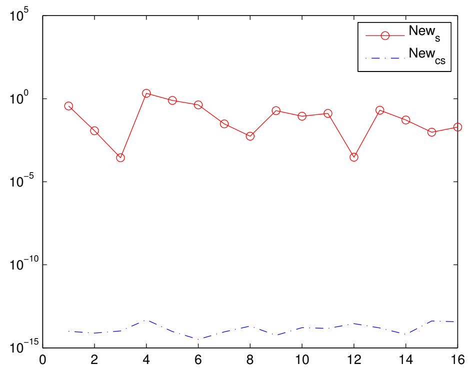

Figure 1: Relative errors of approximation returned by **New_s** and **New_cs** for test matrices as in Example 2 of size $n = 20 + 20 \cdot k$, $k = 1, 2, \ldots, 16$. The figure plots `app_err` versus the integer k.

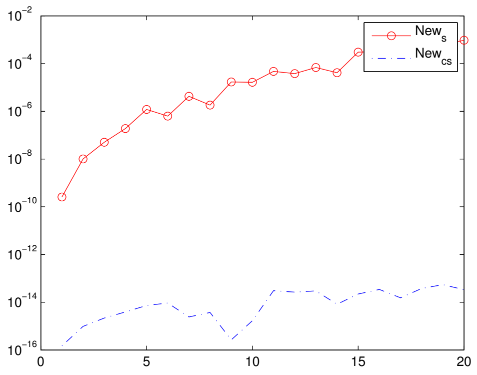

Figure 2: Relative errors of approximation returned by **New_s** and **New_cs** for test matrices as in Example 3 of size $n = 32 + 20 \cdot k$, $k = 1, 2, \ldots, 20$. The figure plots `app_err` versus the integer k.

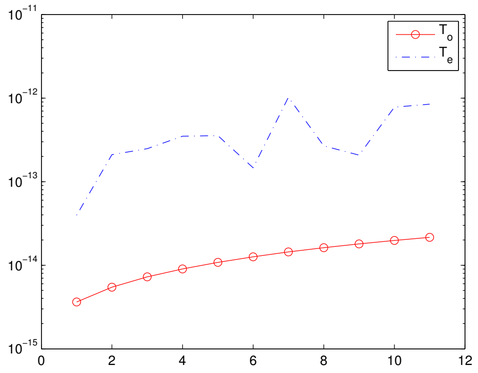

Figure 3: Relative approximation errors returned by **New_cs** for Wilkinson's type matrices of size $n = \{21, 32\} + 20 \cdot k$, $k = 1, 2, \ldots, 11$. The figure plots `app_err` versus the integer k.

Each matrix T_o and T_e has two small eigenvalues of order 1.0e−6 and 1.0e−10, respectively. In Figure 3 we show the plot of the approximation error generated by **New_cs**.

However for harder tests algorithm **New_cs** can even fail. The matrix $T_e = \texttt{wilkinson}(62) - 18.5 * \texttt{eye}(62)$ has two eigenvalues of order 1.0e−16 and **New_cs** fails to resolve the coalescence of these eigenvalues. It is found that the condition number of computing the determinant of T_e is about 1.0e34 and therefore quadruple precision is not enough.

Finally we consider tridiagonal matrices with zero diagonal generally used to check the behavior of algorithms for computing the SVD of bidiagonal matrices. For the matrix T of order 48 with $\alpha = 2^{-10}$ and $\beta = 1$ the function `eig` returns one positive and one negative eigenvalue of order 1.0e−16. Algorithm **New_cs** starts to refine the positive root and after 179 iterations it finds a new approximation of order 1.0e−73 having the approximation error of the same order of the machine precision. The Newton method exhibits a quadratic convergence just only in the last three or four iterations. For most of the steps the iteration shows a linear convergence since the two very close roots are seen as one single multiple root lying at the origin. This makes the refinement process not competitive with faster matrix–based methods [8] which may converge in a few iterations. In Figure 4 we illustrate the convergence history by plotting the relative error $\texttt{rel\_err}: = |\xi_{j+1} - \xi_j|/|\xi_j|$, $1 \leq j \leq 178$, between two consecutive iterates.

For the input matrix T of order 64 with $\alpha = 1$ and $\beta = 256$ our algorithm returns after 194 iterations the approximation $\sigma_{32} = 2.210\,825\,415\,070\,759\text{e}{-75}$ which is correct to full machine precision. However, it is worth noticing that for tridiagonals with zero diagonal there are no appreciable differences between **New_cs** and **New_s**. This is in accordance with the classical rule of thumb that the forward error is of order of the backward error multiplied by the conditioning of the problem. From [6] we know that the problem is almost perfectly well

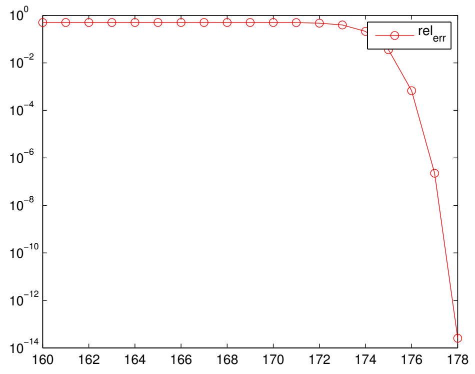

Figure 4: Relative errors between two consecutive iterates at the end of the refinement iterative process. The figure plots `rel_err` versus the iteration index.

conditioned and, moreover, the three-term recurrence computation is backward stable.

4. Conclusion and future work

We have presented a compensated scheme using error free transformations for evaluating the characteristic polynomial and its first derivative of a symmetric tridiagonal matrix. This scheme has been used in conjunction with Newton's method to get a better approximation of matrix eigenvalues. Numerical experiments show that the resulting algorithm generally makes it possible to refine a good initial guess to yield an eigenvalue approximation as accurate as if computed with twice the working precision.

Future work will be directed towards a generalization of the results for the case where the customary Newton method is replaced by the Ehrlich-Aberth iteration for simultaneous eigenvalue computation. This iteration can still be derived as a Newton method applied to a sequence of rational functions depending on the simultaneously computed approximations of the eigenvalues. The application of the resulting algorithm for the solution of the bidiagonal SVD computation is also an ongoing research.

Funding

This work was partially supported by GNCS-INDAM and University of Pisa.

[1] R. Barrio. Rounding error bounds for the Clenshaw and Forsythe algorithms for the evaluation of orthogonal polynomial series. *J. Comput. Appl. Math.*, 138(2):185–204, 2002.

[2] D. A. Bini and G. Fiorentino. Design, analysis, and implementation of a multiprecision polynomial rootfinder. *Numer. Algorithms*, 23(2-3):127–173, 2000.

[3] D. A. Bini and V. Noferini. Solving polynomial eigenvalue problems by means of the Ehrlich-Aberth method. *Linear Algebra Appl.*, 439(4):1130–1149, 2013.

[4] C. Bossut. *A general history of mathematics from the earliest times to the middle of the eighteenth century. Tr. from the French of John Bossut ... To which is affixed a chronological table of the most eminent mathematicians.* London : J. Johnson, 1803.

[5] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math.*, 18:224–242, 1971/72.

[6] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.*, 11(5):873–912, 1990.

[7] D. Elliott. Error analysis of an algorithm for summing certain finite series. *J. Austral. Math. Soc.*, 8:213–221, 1968.

[8] K. V. Fernando and B. N. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67(2):191–229, 1994.

[9] W. Gander and P. Gonnet. Polynomial recurrence, Newton correction and polynomial fractions. Unpublished.

[10] S. Graillat. Accurate simple zeros of polynomials in floating point arithmetic. *Comput. Math. Appl.*, 56(4):1114–1120, 2008.

[11] S. Graillat. Accurate computation of the determinant of a tridiagonal matrix. Personal Communication, 2015.

[12] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix Anal. Appl.*, 16(1):79–92, 1995.

[13] N. Hale and A. Townsend. Fast and accurate computation of Gauss-Legendre and Gauss-Jacobi quadrature nodes and weights. *SIAM J. Sci. Comput.*, 35(2):A652–A674, 2013.

[14] P. Henrici. *Applied and computational complex analysis. Vol. 1.* Wiley Classics Library. John Wiley & Sons, Inc., New York, 1988. Power series—integration—conformal mapping—location of zeros, Reprint of the 1974 original, A Wiley-Interscience Publication.

[15] E. R. Jessup and D. C. Sorensen. A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix Anal. Appl.*, 15(2):530–548, 1994.

[16] H. Jiang, S. Graillat, C. Hu, S. Li, X. Liao, L. Cheng, and F. Su. Accurate evaluation of the k-th derivative of a polynomial and its application. *J. Comput. Appl. Math.*, 243:28–47, 2013.

[17] D. E. Knuth. *The art of computer programming. Vol. 2.* Addison-Wesley, Reading, MA, 1998. Seminumerical algorithms, Third edition [of MR0286318].

[18] P. Langlois. Automatic linear correction of rounding errors. *BIT*, 41(3):515–539, 2001.

[19] T. Y. Li, N. H. Rhee, and Z. G. Zeng. An efficient and accurate parallel algorithm for the singular value problem of bidiagonal matrices. *Numer. Math.*, 69(3):283–301, 1995.

[20] G. Meurant and A. Sommariva. Fast variants of the Golub and Welsch algorithm for symmetric weight functions in Matlab. *Numer. Algorithms*, 67(3):491–506, 2014.

[21] Isaac Newton. *The mathematical papers of Isaac Newton. Vol. VIII.* Cambridge University Press, Cambridge, 1981. 1697–1722, Edited by D. T. Whiteside, With the assistance of A. Prag.

[22] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM J. Sci. Comput.*, 26(6):1955–1988, 2005.

[23] F. Tisseur. Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 22(4):1038–1057 (electronic), 2001.

[24] C. Trefftz, C. C. Huang, P. K. McKinley, T.-Y. Li, and Z. Zeng. A scalable eigenvalue solver for symmetric tridiagonal matrices. *Parallel Comput.*, 21(8):1213–1240, 1995.

[25] J. H. Wilkinson. *The algebraic eigenvalue problem.* Monographs on Numerical Analysis. The Clarendon Press, Oxford University Press, New York, 1988. Oxford Science Publications.

[26] J. Zhang. The scaled Sturm sequence computation. In *Proceedings of the Eighth SIAM Conference on Applied Linear Algebra.* The College of William and Mary, Williamsburg, VA, USA, SIAM, July 2003.