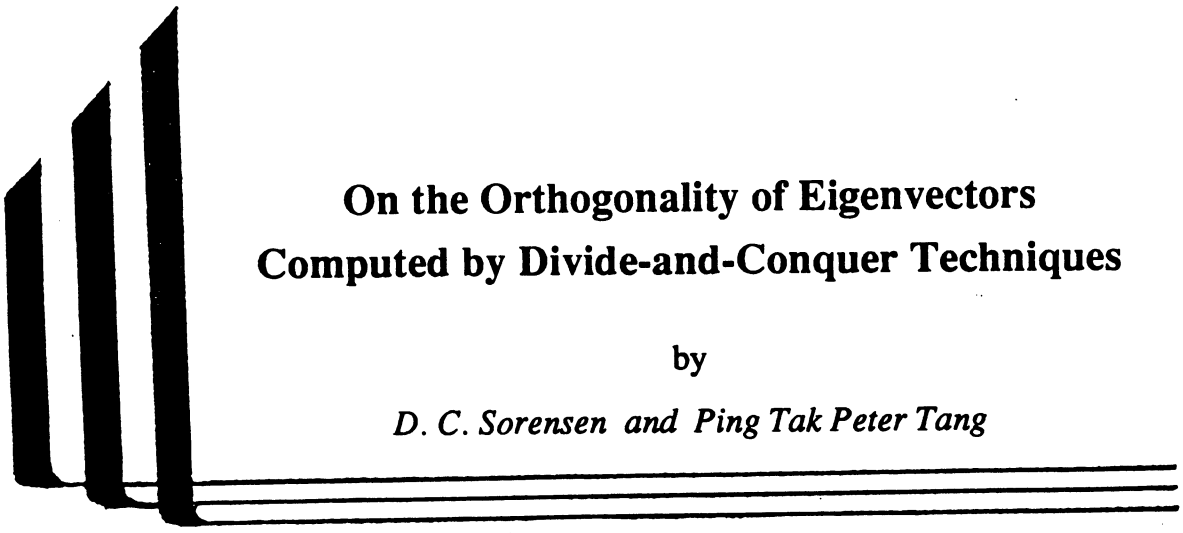
On the Orthogonality of Eigenvectors Computed by Divide-and-Conquer Techniques

Dan Sorensen
P. Tang

CRPC-TR90055
May, 1990

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

PREPRINT MCS-P152-0490



On the Orthogonality of Eigenvectors Computed by Divide-and-Conquer Techniques

by

D. C. Sorensen and Ping Tak Peter Tang

May 1990

Mathematics and Computer Science Division

Argonne National Laboratory

On the Orthogonality of Eigenvectors Computed by Divide-and-Conquer Techniques

D. C. Sorensen*
Department of Mathematical Sciences
Rice University
Houston, Texas 77251-1829

Ping Tak Peter Tang†
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Ave.
Argonne, IL 60439-4801

May 3, 1990

Abstract

A detailed analysis on the accuracy issues in calculating the eigensystems of rank-1 perturbed diagonal systems is presented. Such calculations are the core of the divide-and-conquer technique proposed in [4]. In particular, we prove that the computed eigenvectors are guaranteed orthogonality provided the secular equation is evaluated in a precision that doubles the working one. An efficient algorithm that simulates such "doubled precision" in working precision is also provided. Numerical results that confirm our analysis and implementation are presented.

AMS classification: Primary 65F15, secondary 65G05.

Key words and phrases: Divide-and-conquer, rank-1 update, simulated extra precision.

*This author's work was supported in part the IBM Scientific Center in Bergen (June-July 1988) and by NSF cooperative agreement CCR-8809615.

†This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under Contract W-31-109-Eng-38.

1 Introduction

The symmetric eigenvalue problem is fundamental in computational mathematics. In [4], a new parallel algorithm was presented for the symmetric tridiagonal eigenvalue problem. A surprising result of the work in [4] was that the parallel algorithm developed there, even when run in serial mode, is significantly faster than the previously best sequential algorithm on large problems and is effective on problems of order as moderate as 30.

In [4] the difficulty of computing numerically orthogonal eigenvectors was discussed but not fully resolved. Indeed, for some specially contrived examples, the implementation in [4] fails to deliver fully orthogonal eigenvectors. This paper presents recent progress that resolves those numerical difficulties completely.

We present two approaches. One is to efficiently simulate extra precision at one small critical place within the algorithm. Our analysis shows that such an approach guarantees numerical orthogonality. Moreover, the simulation is straightforward in clean arithmetic environments, most notably those conforming to IEEE Standard 754 [8]. Because the implementation of such simulations may lack portability, it is reasonable to ask whether we can avoid simulation altogether. Along this line, we propose a second approach which, in effect, recasts the critical calculation to avoid some (but not all) roundoff difficulties. Although the second approach is superior to the original method used in [1] and [4], unlike the simulation approach, it cannot guarantee orthognality always (see Section 7 for examples). Thus, we recommend this second approach only to those who wish to avoid subtle use of floating-point arithmetic even in the expense, in this case, of guaranteed accuracy.

The rest of the paper is organized as follows. Section 2 reviews the core of the algorithm presented in [4]. Section 3 discusses the basic requirement to maintain numerical orthogonality. Section 4 analyzes the accuracy of root finders in general. That analysis is the basis of the two approaches presented in Sections 5 and 6. Section 7 presents results of some numerical experiments illustrating the approaches and confirming our claims. Section 8 compares our work with the independent work of Kahan. Section 9 gives some concluding remarks and briefly discusses computer arithmetic and language issues related to implementing our simulation algorithm as a library routine.

2 The Divide-and-Conquer Scheme

The problem we consider is the following: Given a real $n \times n$ symmetric matrix A, find all of the eigenvalues and corresponding eigenvectors of A. It is well known [13] that under these assumptions

$$A = QDQ^T, \quad \text{with} \quad Q^TQ = I, \tag{2.1}$$

so that the columns of the matrix Q are the orthonormal eigenvectors of A, and D is the diagonal matrix of eigenvalues. The standard algorithm for computing this decomposition is first to use a finite algorithm to reduce A to tridiagonal form by a sequence of

Householder transformations, and then to apply a version of the QR algorithm to obtain all the eigenvalues and eigenvectors of the tridiagonal matrix [13]. Once this tridiagonal form has been obtained, the algorithm described in [4], instead of QR, may be used to find the eigenvalues and eigenvectors in parallel. The method given in [4] is based on a divide-and-conquer algorithm suggested by Cuppen [2]. A fundamental tool used to implement this algorithm is a method developed by Bunch, Nielsen, and Sorensen [1] for updating the eigensystem of a symmetric matrix after modification by a rank-one change. This rank-one updating method was inspired by some earlier work of Golub [7] on modified eigenvalue problems. The basic idea of the new method is to use rank-one modifications to tear out selected off-diagonal elements of the tridiagonal problem in order to introduce a number of independent subproblems of smaller size. The subproblems are solved at the lowest level by using the subroutine TQL2 from EISPACK [10], and then the results of these problems are successively glued together by using the routine SESUPD that was developed based on the ideas in [1]. The details of the algorithm and implementation are presented in [4] and [5].

The crux of the algorithm is to divide a given problem into two smaller subproblems. To do this, we consider the symmetric tridiagonal matrix

$$\begin{aligned} T &= \begin{pmatrix} T_1 & \beta e_k e_1^T \\ \beta e_1 e_k^T & T_2 \end{pmatrix} \\ &= \begin{pmatrix} \hat{T}_1 & \\ & \hat{T}_2 \end{pmatrix} + \vartheta\beta \begin{pmatrix} e_k \\ \frac{1}{\vartheta} e_1 \end{pmatrix} \begin{pmatrix} e_k^T & \frac{1}{\vartheta} e_1^T \end{pmatrix}, \end{aligned}$$

where $1 \le k \le n$ and e_j represents the j-th unit vector of appropriate dimension. The k-th diagonal element of T_1 has been modified to give $\hat{T}_1$, and the first diagonal element of T_2 has been modified to give $\hat{T}_2$. Potential numerical difficulties associated with cancellation may be avoided through the appropriate choice of ϑ. If the diagonal entries to be modified are of the same sign, then $\vartheta = \pm 1$ is chosen so that $-\vartheta\beta$ has this sign and cancellation is avoided. If the two diagonal entries are of opposite sign, then the sign of ϑ is chosen so that $-\vartheta\beta$ has the same sign as one of the elements, and the magnitude of ϑ is chosen to avoid severe loss of significant digits when β/ϑ is subtracted from the other. This is perhaps a minor detail, but it does allow the partitioning to be selected solely on the basis of position and without regard to numerical considerations.

Now we have two smaller tridiagonal eigenvalue problems to solve. According to Equation 2.1, we compute the two eigensystems

$$\hat{T}_1 = Q_1 D_1 Q_1^T, \qquad \hat{T}_2 = Q_2 D_2 Q_2^T.$$

This gives

$$\begin{aligned} T &= \begin{pmatrix} Q_1 D_1 Q_1^T & \\ & Q_2 D_2 Q_2^T \end{pmatrix} + \vartheta\beta \begin{pmatrix} e_k \\ \frac{1}{\vartheta} e_1 \end{pmatrix} \begin{pmatrix} e_k^T & \frac{1}{\vartheta} e_1^T \end{pmatrix}, \\ &= \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \left\{ \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \vartheta\beta \begin{pmatrix} q_1 \\ \frac{1}{\vartheta} q_2 \end{pmatrix} \begin{pmatrix} q_1^T & \frac{1}{\vartheta} q_2^T \end{pmatrix} \right\} \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix}^T, \end{aligned}$$

where $q_1 = Q_1^T e_k$ and $q_2 = Q_2^T e_1$. The problem at hand now is to compute the eigensystem of the interior matrix in this equation.

The general problem we are required to solve is that of computing the eigensystem of a matrix of the form

$$\hat{Q}\hat{D}\hat{Q}^T = D + \rho zz^T,$$

where D is a real $n \times n$ diagonal matrix, ρ is a nonzero scalar, and z is a real vector of order n. It is assumed without loss of generality that z has Euclidean norm 1.

We seek a formula for an eigenpair for the matrix $D + \rho zz^T$. In [4] such a formula is derived under the assumptions that $D = \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_n)$ with $\delta_1 < \delta_2 < \ldots < \delta_n$ and that no component ζ_i of the vector z is zero. Moreover, the deflation process described in [4] also guarantees that $|\zeta_j| \geq \varepsilon$ where ε is the unit roundoff of the arithmetic precision in question. We reproduce the key formulas here: If λ is a root of the equation

$$1 + \rho z^T (D - \lambda I)^{-1} z = 0$$

and

$$q = \alpha (D - \lambda I)^{-1} z,$$

then (q, λ) is an eigenpair; i.e., it satisfies the relation

$$D + \rho zz^T q = \lambda q.$$

The scalar α may be chosen so that $\|q\| = 1$ to obtain an orthonormal eigensystem.

If we write the equation

$$1 + \rho z^T (D - \lambda I)^{-1} z = 0$$

in terms of the components ζ_j's of z, then λ must be a root of the equation

$$f(\lambda) = 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{\delta_j - \lambda}. \tag{2.2}$$

This equation is usually known as the secular equation (see [7]). Under our assumptions, this equation has precisely n roots, one in each of the open intervals (δ_j, δ_{j+1}), $j = 1, 2, \ldots, n-1$, and one to the right of δ_n if $\rho > 0$, or one to the left of δ_1 if $\rho < 0$. For our purpose, we can assume without loss of generality that $\rho > 0$ and thus simplify the presentation, knowing that the n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ satisfy

$$\delta_1 < \lambda_1 < \delta_2 < \lambda_2 < \ldots < \delta_n < \lambda_n < \delta + \rho.$$

(The last inequality is easy to derive from Equation 2.2.) As soon as any of the eigenvalues is found, we can construct the corresponding eigenvector by the formula

$$q = \alpha (D - \lambda I)^{-1} z.$$

An excellent numerical method was developed in [1] to find the roots of the secular equation and, as a by-product, to compute the eigenvectors. Even though this method is satisfactory for most problems and always delivers accurate eigenvalues, the computed eigenvectors may lose orthogonality when there are extremely difficult situations in the secular equation. What exactly contributes to difficult situations and how to cope with them are the subjects of the rest of this paper.

3 Orthogonality of Vectors

The first result on the orthogonality of the computed eigenvectors is a lemma proved in [4]:

Lemma 1 *Let*

$$q_\lambda^T \equiv \left(\frac{\zeta_1}{\delta_1-\lambda}, \frac{\zeta_2}{\delta_2-\lambda}, \ldots, \frac{\zeta_n}{\delta_n-\lambda}\right) \cdot \sqrt{\rho/f'(\lambda)}$$

for $\lambda \notin \{\delta_1, \delta_2, \ldots, \delta_n\}$. *Then for* λ *and* $\mu \notin \{\delta_1, \delta_2, \ldots, \delta_n\}$,

$$q_\lambda^T q_\mu = \frac{f(\lambda)-f(\mu)}{(\lambda-\mu)\sqrt{f'(\lambda)f'(\mu)}}.$$

Note that q_λ in Lemma 1 is always of unit length, and the set of n vectors selected by setting λ to the n roots of the secular equation $f(\lambda) = 0$ is the set of eigenvectors for $D + \rho zz^T$. Moreover, the lemma shows that the eigenvectors for $D + \rho zz^T$ are mutually orthogonal (as expected). Finally, the term $\lambda - \mu$ appearing in the denominator sends up a warning that it may be difficult to attain orthogonal eigenvectors when the roots λ and μ are close. The key result on maintaining orthogonality numerically is given by the next lemma, which was proved in [4].

Lemma 2 *Let* $\hat{q}_\lambda = \hat{u}_\lambda/\|\hat{u}_\lambda\|, \hat{q}_\mu = v_\mu/\|v_\mu\|$ *where*

$$\hat{u}_\lambda^T = \left(\frac{\zeta_1}{\Delta_1}, \frac{\zeta_2}{\Delta_2}, \ldots, \frac{\zeta_n}{\Delta_n}\right), \quad \text{and} \quad \hat{v}_\mu^T = \left(\frac{\zeta_1}{\Lambda_1}, \frac{\zeta_2}{\Lambda_2}, \ldots, \frac{\zeta_n}{\Lambda_n}\right)$$

be the computed eigenvectors corresponding to q_λ *and* q_μ. *Let*

$$\Delta_j = (\delta_j - \lambda)(1+\vartheta_j), \quad \text{and} \quad \Lambda_j = (\delta_j - \mu)(1+\eta_j)$$

for $j = 1, 2, \ldots, n$. *If* $|\vartheta_j|, |\eta_j| \le \gamma \ll 1$. *Then*

$$|\hat{q}_\lambda^T \hat{q}_\mu| \le \gamma(2+\gamma)\left(\frac{1+\gamma}{1-\gamma}\right)^2.$$

Lemma 2 shows that numerical orthogonality can be assured whenever it is possible to compute the distances $\delta_j - \lambda$, $j = 1, 2, \ldots, n$, to a high relative precision.

Indeed, the common goal of the two approaches we take is to accurately compute $\delta_j - \lambda_i$, $j = 1, 2, \ldots, n$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are the n eigenvalues of $D + \rho zz^T$. Before we present our two schemes for tackling the problem, we first examine the accuracy issue of a general iterative root finder.

4 Accuracy of Roots Computed by Iterative Solvers

How accurately can a root x^* of a certain equation $g(x) = 0$ be computed by some iterative scheme such as, say, Newton iteration? This question is germane because the differences $\delta_j - \lambda_i$ that we seek are typically characterized as roots of some equation, such as the secular equation. Since we are interested in computing the differences $\delta_j - \lambda_i$ to high relative precision, we now investigate under what condition an iterative root finder can locate a root to high relative precision.

A common scenario of iterative root finders is as follows. Let $x^{(k)}$ be the approximate root at the k-th iteration. To advance to the next iteration, the scheme prescribes a correction term $\tau^{(k)}$ so that $x^{(k)} + \tau^{(k)}$ is an improved approximate root. To continue the iteration, $x^{(k+1)}$ is set to $x^{(k)} + \tau^{(k)}$. In the absence of rounding errors, robust root finders would yield a sequence of $x^{(k)}$ that converges to x^*. In the presence of rounding error, however,

$$\text{computed } \tau^{(k)} \neq \tau^{(k)}.$$

Therefore, in order that the root be determined to nearly full relative accuracy, the number of correct digits in "computed $\tau^{(k)}$" must be roughly equal to the number of digits in $\tau^{(k)}$ that will affect $fl(x^{(k)} + \tau^{(k)})$. Thus we must have

$$\left|\frac{\text{computed } \tau^{(k)} - \tau^{(k)}}{\tau^{(k)}}\right| \cdot \left|\frac{\tau^{(k)}}{x^{(k)}}\right| \leq M \cdot \varepsilon, \tag{4.3}$$

where M is a constant not much bigger than unity, and ε is the relative precision of the machine. For example, if $x^{(k)}$ approximates x^* to 13 digits in a 17-digit environment, then it suffices to have computed $\tau^{(k)}$ accurate to 4 digits. Even though $x^{(k)} + \tau^{(k)}$ may approximate x^* to 26 digits (for a quadratically convergent root finder), having the correct $\tau^{(k)}$ would not further improve the already satisfactory situation, because

$$fl(x^{(k)} + \text{computed } \tau^{(k)}) \approx fl(x^{(k)} + \tau^{(k)}).$$

What, then, determines the accuracy in "computed $\tau^{(k)}$"? In order to have a superlinearly convergent root finder, it is necessary that $\tau^{(k)}$ approaches the Newton step $-g(x^{(k)})/g'(x^{(k)})$ in both length and direction (see [6]). Typically, for such iterations $\tau^{(k)} = -g(x^{(k)})/D^{(k)}$ where $D^{(k)}$ approximates $g'(x^{(k)})$. Assuming $g'(x^*) \neq 0$ implies that, as $x^{(k)}$ approaches x^*, the number of correct digits in "computed $\tau^{(k)}$" is roughly the same as the number of correct digits in "computed $g(x^{(k)})$," the computed value of the function at the k-th approximate root. This means that

$$\left|\frac{\text{computed } \tau^{(k)} - \tau^{(k)}}{\tau^{(k)}}\right| \approx \left|\frac{\text{computed } g(x^{(k)}) - g(x^{(k)})}{g(x^{(k)})}\right|.$$

The situation is a competition. As $x^{(k)}$ approaches x^*, the accuracy requirement on the computed correction relaxes; but the actual accuracy of that computed quantity also

decreases because $g(x^{(k)})$ is small, usually suggesting severe cancellation has taken place in its computation. The indicator on how accurately x^* can be determined is therefore the quantity

$$\frac{1}{\varepsilon}\left|\frac{\text{computed } \tau^{(k)} - \tau^{(k)}}{\tau^{(k)}}\right| \cdot \left|\frac{\tau^{(k)}}{x^{(k)}}\right|$$

which is approximately

$$\frac{1}{\varepsilon}\left|\frac{\text{computed } g(x^{(k)}) - g(x^{(k)})}{g(x^{(k)})}\right| \cdot \left|\frac{\tau^{(k)}}{x^{(k)}}\right|.$$

As $x^{(k)}$ approaches x^*, $|g(x^{(k)})| \approx |\tau^{(k)} g'(x^*)|$. Thus, assuming $|g'(x^*)| > 0$,

$$\begin{aligned}\frac{1}{\varepsilon}\left|\frac{\text{computed } \tau^{(k)} - \tau^{(k)}}{\tau^{(k)}}\right| \cdot \left|\frac{\tau^{(k)}}{x^{(k)}}\right| &\approx \frac{1}{\varepsilon}\left|\frac{\text{computed } g(x^{(k)}) - g(x^{(k)})}{g(x^{(k)})}\right| \cdot \left|\frac{\tau^{(k)}}{x^{(k)}}\right| \\ &\approx \frac{|\text{computed } g(x^{(k)}) - g(x^{(k)})|}{\varepsilon |x^* \cdot g'(x^*)|}.\end{aligned}$$

Hence the crux is to be able to compute $g(x^{(k)})$ so accurately that the quantity

$$\frac{|\text{absolute error in computed } g(x^{(k)})|}{\varepsilon |x^* \cdot g'(x^*)|}$$

is bounded by a moderate constant. If $M \approx 1$, one can expect x^* be computed to nearly full accuracy. If $M \approx 10^3$, say, then "computed x^*" will in general be approximately 3 digits short, unless, of course, if the absolute error in "computed g" is reduced by a factor of 10^3 via, for example, extra-precise arithmetic.

5 The Secular Equation

Consider the eigenvector corresponding to the i-th eigenvalue λ_i. We now reexamine the method proposed in [4] which calculates the differences $\Delta_j^* := \delta_j - \lambda_i$, $j = 1, 2, \ldots, n$, by solving for the root of

$$f(\lambda) = 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{\Delta_j}, \qquad \Delta_j = \delta_j - \lambda = \delta_j - \delta_I - (\lambda - \delta_I) \quad j = 1, 2, \ldots, n,$$

in the interval (δ_i, δ_{i+1}) , where δ_I is the endpoint of this interval which is closest to λ_i . With this definition, our scheme fits exactly into the previous description with $g(x) := f(\delta_I + x)$, where $x := \lambda - \delta_I$. Hence, we must estimate the absolute error in computing f and f's derivative at λ_i. Since

$$\min(|\Delta_i^*|, |\Delta_{i+1}^*|) \leq \min_{1 \leq j \leq n} |\Delta_j^*|,$$

the indicator is thus

$$M \equiv \frac{|\text{absolute error in computed } f|}{\varepsilon \cdot \min(|\Delta_i^*|, |\Delta_{i+1}^*|) \cdot |f'(\lambda_i)|}.$$

Clearly, if the function f is computed in the obvious way, the absolute error will be at least

$$\varepsilon \max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j} \right|$$

in general. Thus, the quantity of interest is

$$\begin{aligned} M &\approx \frac{\varepsilon \max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j} \right|}{\varepsilon \min(|\Delta_i^*|, |\Delta_{i+1}^*|) \left| \rho \sum_{1 \le j \le n} \frac{\zeta_j^2}{\Delta_j^{*2}} \right|} \\ &= \frac{\max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j^*} \right|}{\min(|\Delta_i^*|, |\Delta_{i+1}^*|) \left| \rho \sum_{1 \le j \le n} \frac{\zeta_j^2}{\Delta_j^{*2}} \right|}. \end{aligned}$$

If $|\Delta_i^*| \le |\Delta_{i+1}^*|$ and if $\max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j} \right|$ occurs at index i, then

$$M \approx \frac{\left| \rho \frac{\zeta_i^2}{\Delta_i^*} \right|}{|\Delta_i^*| \left| \rho \sum_{1 \le j \le n} \frac{\zeta_j^2}{\Delta_j^{*2}} \right|} \le 1,$$

implying that Δ_i^* can be computed to high relative precision even when f is evaluated in the most straightforward way. Although one may think that when Δ_i^* is small, the term $|\zeta_i^2/\Delta_i^*|$ would dominate, the "weights" ζ_j^2 may be distributed so that in fact $|\zeta_i^2/\Delta_i^*|$ is nowhere near dominating. Indeed, in the contrived examples, the weights can be distributed so that M is as large as $1/\varepsilon$. As shown in Section 7, the loss of orthogonality exhibited by the implementation in [4] is totally consistent with the analysis here.

Let us now obtain a general estimate for M:

$$M \approx \frac{\max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j^*} \right|}{\min(|\Delta_i^*|, |\Delta_{i+1}^*|) \left| \rho \sum_{1 \le j \le n} \frac{\zeta_j^2}{\Delta_j^{*2}} \right|}.$$

Let

$$\max_{1 \le j \le n} \left| \rho \frac{\zeta_j^2}{\Delta_j^*} \right| = \left| \rho \frac{\zeta_l^2}{\Delta_l^*} \right| \qquad \text{and} \qquad \min_{1 \le j \le n} |\Delta_j^*| = |\Delta_m^*|.$$

If $l = m$, then M is trivially bounded by 1. If $l \neq m$, then

$$\begin{aligned}\sum_{1\le j\le n} \frac{\zeta_j^2}{\Delta_j^{*2}} &\geq \frac{\zeta_l^2}{\Delta_l^{*2}} + \frac{\zeta_m^2}{\Delta_m^{*\,2}} \\ &\geq 2\left|\frac{\zeta_l\zeta_m}{\Delta_l^*\Delta_m^*}\right|.\end{aligned}$$

Thus,

$$\begin{aligned}M &\leq \frac{1}{2}|\zeta_l/\zeta_m| \\ &\leq \frac{1}{2}\max_{1\le l,m\le n}|\zeta_l/\zeta_m|.\end{aligned}$$

The deflation process in [4] guarantees that $|\zeta_j| \geq \varepsilon$ for $j = 1, 2, \ldots, n$. On the other hand, $|\zeta_j| \leq \|z\| = 1$, for $j = 1, 2, \ldots, n$. Thus

$$M \leq \frac{1}{2\varepsilon}.$$

Thus, the analysis shows that if f is evaluated in a way equivalent to carrying double the working precision for all intermediate computations, *the eigenvectors will be guaranteed mutual orthogonality.*

The challenge is, therefore, to evaluate the function values at the iterations as if "doubled precision" were used throughout before the result is rounded back to working precision. The expression of the function values has the form

$$\begin{aligned}f &= 1 + \rho\sum_{j=1}^{n}\frac{\zeta_j^2}{\Delta_j} \\ &= 1 + \rho\sum_{j=1}^{n}\frac{\zeta_j^2}{\Delta_I - (\delta_I - \delta_j)},\end{aligned}$$

where $I \in \{i, i+1\}$ and $|\Delta_I| = \min\{|\Delta_j|\}$. If working precision is single precision and if double precision is available (or when working precision is double and "quad" precision is available), the problem is trivially solved. We therefore address the situation where working precision is double precision and that it is the highest precision available. For simplicity, we assume that the floating-point environment conforms to IEEE Standard 754 [8]. The techniques below, however, also work correctly on VAXes and IBMs.

5.1 Some Extra-Precision Primitives

We introduce several basic extra-precision operations which will be used to evaluate the secular equation accurately in the next subsection. A natural data type for simulating a precision that doubles the working one is a pair of working-precision variables. Therefore,

we call a pair of working-precision numbers (X, x) to be in simulated double precision (SDP) if

$$|x| \leq 10 \cdot \varepsilon |X|.$$

The idea is that the mathematical sum (i.e., not evaluated in computer arithmetic) $X + x$ is a number having twice the number of significant bits than working precision can offer. The constant "10" is more or less arbitrary and is chosen here for ease of presentation.

The first primitive has to do with getting an SDP sum of two working-precision numbers. The technique is standard (see [3] for example).

Algorithm 1 *DP_Add2(A, B : working precision) return SDP.*

1. $X := A + B$
2. If $|A| < |B|$, swap A and B (so $|A| \geq |B|$ always).
3. $x := (A - X) + B$; must observe parentheses.
4. Return (X, x).

The next primitive generalizes DP_Add2 to sum a SDP number and a working-precision number.

Algorithm 2 *DP_Add3 ((X,x):SDP, Y: working precision) return SDP.*

1. $(S, s) :=$ DP_Add2(X, Y).
2. $T := s + x$.
3. $(Z, z) :=$ DP_Add2(S, T).
4. Return (Z, z).

Finally, we need a primitive that produces an SDP quotient of a working-precision number divided by an SDP number $A/(X, x)$. The method first approximates $A/(X + x)$ to working precision by $Y := A/X$. Then, the "correction term" $(A/(X + x)) - Y$ is carefully calculated to working precision.

Algorithm 3 *DP_Div (A: working precision, (X, x): SDP) return SDP.*

1. $Y := A/X$.
2. $X_1 :=$ *round-to-double*(*round-to-single*(X)), $X_2 := X - X_1$. See the notes below.
3. Split Y into Y_1, Y_2 similarly.

4. Form $(X_1 + X_2)(Y_1 + Y_2)$ carefully by

$$\begin{aligned} A_1 &:= X_1 * Y_1 \\ A_2 &:= X_1 * Y_2 \\ A_3 &:= X_2 * Y_1 \\ A_4 &:= X_2 * Y_2 \end{aligned}$$

The idea is that A_1, A_2, and A_3 are all calculated exactly.

5. $t := x * Y$.

6. $(S, s) :=$ DP_Add2$(A2, A3)$.

7. $(S, s) :=$ DP_Add3$((S, s), A4)$.

8. $Z := (((A - A_1) - S) - s) - t \ldots$ Observe order!

9. $y := Z/X$.

10. Return (Y, y).

Notes: Step 2 of DP_Div needs some elaboration. The idea is that X has to be decomposed in a way that the leading part X_1 has slightly fewer than half the significant bits of working precision. This ensures that both $X_1 * X_1$ and $X_1 * X_2$ are computable exactly in working precision. Since our use of DP_Div involves numbers that are not extreme in magnitude, the "round-to-single" operation is convenient to use and would not suffer intermediate over/underflow. An alternative to the decomposition when the IEEE 754-recommended functions "logb" and "scalb" are available is the following: With all intermediate calculations rounded to working precision, compute

$$\begin{aligned} T &:= \text{sign}(X) \cdot \text{scalb}(\text{logb}(|X|) + 28, 1), \\ X_1 &:= (T + X) - T, \quad \text{and} \\ X_2 &:= X - X_1. \end{aligned}$$

5.2 Extra-Precise Evaluation of the Secular Equation

We now describe how the expression

$$f = 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{\Delta_I - (\delta_I - \delta_j)}$$

can be calculated as if all intermediate computations are carried out in SDP. First compute in working precision $B := 1/\rho$ and $A_j := \zeta_j * \zeta_j$, $j = 1, 2, \ldots, n$. Then it suffices to calculate

$$\rho * \left(B + \sum_{j=1}^{n} \frac{A_j}{\Delta_I - (\delta_I - \delta_j)} \right)$$

accurately because this is tantamount to perturbing ρ and ζ_j's in the last bit prior to solving the rank-1 update problem in question.

Algorithm 4 *DP_Sec ($A_j's, B, \Delta_I, \delta_j's, \rho$: working precision) return working precision*

1. $\Phi := B$, $\phi := 0$, $\Psi := 0$, $\psi := 0$.
2. For $j = 1, 2, \ldots, n$ do
   - $(P_1, P_2) :=$ DP_Add2$(-\delta_I, \delta_j)$.
   - $(X, x) :=$ DP_Add3$((P_1, P_2), \Delta_I)$.
   - $(Y, y) :=$ DP_Div$(A_j, (X, x))$
   - If $j \leq i$ then
     - $(\Psi, Y) :=$ DP_Add2(Ψ, Y).
     - $\psi := \psi + Y + y$.
   - else
     - $(\Phi, Y) :=$ DP_Add2(Φ, Y).
     - $\phi := \phi + Y + y$
   - End if.
3. End do.
4. $Z := \rho * ((\Phi + \Psi) + \phi + \psi) \ldots$ Observe order!
5. Return Z.

It is important to note that in the whole course of computing the i-th eigenpair, DP_Sec needs only be invoked once. The reason is that

$$\begin{aligned} \text{new } f &= 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{\Delta_j - \tau} \\ &= \text{current } f + \rho\tau \sum_{j=1}^{n} \frac{\zeta_j^2}{\Delta_j(\Delta_j - \tau)} \\ &= \text{current } f + \tilde{f}(\tau). \end{aligned}$$

Note that $\tilde{f}$ can be easily computed to full working precision (without simulation). Thus the iterative root finder can base its iteration on straightforwardly computed f until

$$|f| \approx |\text{absolute error in computed } f|.$$

At that time, DP_Sec is invoked. Subsequent function values are obtained by

$$\text{new } f = \text{current } f + \tilde{f}(\text{new } \tau).$$

This computation needs no extra precision.

6 Reformulation of the Secular Equation

In this section we characterize the i-th eigenvalue as a root of another equation $g(\lambda) = 0$ such that, most of the time, working-precision evaluation suffices for orthogonality's sake. From Lemma 1, we have

$$v_\lambda^T v_\mu = \frac{f(\lambda) - f(\mu)}{(\lambda - \mu) \cdot \rho}, \qquad \lambda, \mu \notin \{\delta_1, \ldots, \delta_n\},$$

where

$$v_x^T = \left(\frac{\zeta_1}{\delta_1 - x}, \frac{\zeta_2}{\delta_2 - x}, \ldots, \frac{\zeta_n}{\delta_n - x} \right),$$

and

$$f(x) = 1 + \rho \sum_{j=1}^{n} \frac{\zeta_j^2}{\delta_j - x}.$$

Thus, if the $(i-1)$-th eigenvalue $\lambda_{i-1} \in (\delta_{i-1}, \delta_i)$ is known, the i-th eigenvalue is characterized by

$$g(\lambda) = 0, \qquad \lambda \in (\delta_i, \delta_{i+1}),$$

where

$$g(x) = v_x^T v_{\lambda_{i-1}}.$$

To predict whether such a reformulation helps maintain orthogonality of eigenvectors, we apply the analysis in Section 3:

$$g(x) = \sum_{j=1}^{n} \frac{\eta_j}{\Delta_j}, \text{ where } \eta_j = \frac{\zeta_j^2}{\delta_j - \lambda_{i-1}}, \text{ and } \Delta_j = \delta_j - x.$$

At the root λ_i, $g(\lambda_i) = 0$, $\frac{\eta_i}{\Delta_i^*} < 0$, and $\frac{\eta_j}{\Delta_j^*} > 0$ for $j \neq i$ where $\Delta_j^* = \delta_j - \lambda_i$. Thus

$$\left| \frac{\eta_i}{\Delta_i^*} \right| \geq \left| \frac{\eta_j}{\Delta_j^*} \right| \text{ for } j \neq i.$$

Consequently, at $x \approx \lambda_i$, $g(x)$ evaluated in the straightforward manner would have an absolute error equal approximately to

$$\varepsilon \cdot \left| \frac{\eta_i}{\Delta_i^*} \right|.$$

Next, we would have to estimate the derivative g' at λ_i:

$$g'(\lambda_i) = \sum_{j=1}^{n} \frac{\eta_j}{\Delta_j^{*2}}.$$

Lemma 3 *Let* $|\lambda_i - \delta_i| = \alpha|\lambda_i - \delta_{i-1}|$, $(0 < \alpha < 1)$. *Then*

$$|g'(\lambda_i)| \geq (1-\alpha)\left|\frac{\eta_i}{\Delta_i^*}\right|.$$

Proof. Let $\Delta_j^* = \delta_j - \lambda_i$, $j = 1, 2, \ldots, n$.

$$\begin{aligned} g'(\lambda_i) &= \sum_1^n \frac{\eta_j}{\Delta_j^{*2}} \\ &= \left(\sum_{j\neq i} \frac{1}{\Delta_j^*} \cdot \frac{\eta_j}{\Delta_j^*}\right) + \frac{1}{\Delta_i^*}\frac{\eta_i}{\Delta_i^*} \\ &= \sum_{j\neq i} \frac{1}{\Delta_j^*}\frac{\eta_j}{\Delta_j^*} + \frac{-1}{\Delta_i^*}\sum_{j\neq i}\frac{\eta_j}{\Delta_j^*} \\ &= \sum_{j\neq i}\left(\frac{1}{\Delta_j^*} - \frac{1}{\Delta_i^*}\right)\frac{\eta_j}{\Delta_j^*} \\ &= \sum_{j\neq i}\frac{(\delta_i - \delta_j)}{\Delta_j^*\Delta_i^*}\frac{\eta_j}{\Delta_j^*}. \end{aligned}$$

Note that $\frac{\delta_i - \delta_j}{\Delta_j^*\Delta_i^*} > 0$ and $\frac{\eta_j}{\Delta_j^*} > 0$ for $j \neq i$; thus $g'(\lambda_i) > 0$ and

$$g'(\lambda_i) = \sum_{j\neq i}\left|\frac{\delta_i - \delta_j}{\Delta_j^*\Delta_i^*}\right|\left|\frac{\eta_j}{\Delta_j^*}\right|.$$

For $j > i$,

$$\left|\frac{\delta_i - \delta_j}{\Delta_j^*}\right| = \left|\frac{\delta_j - \delta_i}{\delta_j - \lambda_i}\right| \geq 1.$$

For $j < i$,

$$\begin{aligned} \frac{\delta_i - \delta_j}{\Delta_j^*} &= 1 - \frac{\lambda_i - \delta_i}{\lambda_i - \delta_j} \\ &\geq 1 - \frac{\lambda_i - \delta_i}{\lambda_i - \delta_{i-1}} \\ &\geq 1 - \alpha. \end{aligned}$$

Hence

$$\begin{aligned} g' &\geq \frac{(1-\alpha)}{|\Delta_i^*|}\sum_{j\neq i}\frac{\eta_j}{\Delta_j} \\ &\geq \frac{(1-\alpha)}{|\Delta_i^*|}\left|\frac{\eta_i}{\Delta_i^*}\right|. \end{aligned}$$

■

With the bound in Lemma 3, we have

$$M \leq \frac{|\Delta_i^*|(1-\alpha)^{-1}}{\min_{1\leq j\leq n}\{|\Delta_j^*|\}}.$$

Hence, whenever $|\lambda_i - \delta_i|$ is not too large compared to $|\lambda_i - \delta_{i-1}|$, Δ_i can be computed to nearly full accuracy without any need of extra-precision arithmetic. Lemma 3 helps us evaluate the potential of the reformulation; it also guides us in creating challenging examples (see Section 7). Note that the quantity $|\delta_i - \delta_{i+1}|/|\delta_i - \delta_{i-1}|$, which is known a priori, can be used instead of $|\lambda_i - \delta_i|/|\lambda_i - \delta_{i-1}|$; although this a priori bound is not as sharp.

The analysis shows that Δ_i^* can be accurately obtained whenever α is reasonably below 1, say, ≤ 0.9. Therefore, orthogonality can be assured provided $|\Delta_i^*| \leq |\Delta_{i+1}^*|$ or provided $i = n$. What happens if $|\Delta_{i+1}^*| \ll |\Delta_i^*|$ and $i \leq n-1$? The bound in Lemma 3 is not helpful in this case. We turn to our next estimate of $g'(\lambda_i)$.

Lemma 4 *For $i \leq n-1$,*

$$|g'(\lambda_i)| \geq \frac{2}{\lambda_i - \lambda_{i-1}} \left| \frac{\zeta_i \zeta_{i+1}}{\Delta_i^* \Delta_{i+1}^*} \right|.$$

Proof. Since

$$\begin{aligned} g(x) &= \frac{f(x) - f(\lambda_{i-1})}{(x-\lambda_{i-1})\cdot\rho} \\ &= \frac{f(x)}{(x-\lambda_{i-1})\rho}, \end{aligned}$$

therefore

$$\begin{aligned} g'(\lambda_i) &= \frac{f'(\lambda_i)}{(\lambda_i - \lambda_{i-1})\rho} \\ &= \frac{1}{\lambda_i - \lambda_{i-1}} \sum_{j=1}^{n} \frac{\zeta_j^2}{(\delta_j - \lambda_i)^2} \\ &\geq \frac{2}{\lambda_i - \lambda_{i-1}} \left| \frac{\zeta_i \zeta_{i+1}}{\Delta_i^* \Delta_{i+1}^*} \right|. \end{aligned}$$

■

With the new estimate, if $i \leq n-1$ and $|\Delta_{i+1}^*| < |\Delta_i^*|$, we have

$$\begin{aligned} M &\leq \frac{\left|\frac{\eta_i}{\Delta_i^*}\right|}{|\Delta_{i+1}^*| \cdot |g'(\lambda_i)|} \\ &\leq \left| \frac{\lambda_i - \lambda_{i+1}}{\delta_i - \lambda_{i-1}} \right| \cdot \left| \frac{\zeta_i}{\zeta_{i+1}} \right|. \end{aligned}$$

This analysis shows that as long as δ_i is not too much closer to λ_{i-1} than to λ_i and that $|\zeta_i/\zeta_{i+1}|$ is moderate, the scheme is able to deliver Δ_{i+1}^* to high relative accuracy.

Unlike solving for the root of the secular equation with function evaluation by DP_Sec which guarantees orthogonality, solving $g(x) = 0$ is effective only some of the time. The latter approach, however, has the advantage that, in some practical situations, it is able to maintain orthogonality where otherwise extra-precision simulation would be needed, should we try to solve the secular equation.

6.1 A Root Finder for $g(x)$

We now present a quadratically convergent root finder for the reformulated equation

$$g(x) \equiv v_x^T v_{\lambda_{i-1}} = 0.$$

The root finder is based on rational interpolation, similar to the root finder in [1].

Let $\lambda \in (\delta_i, \delta_{i+1})$ be an approximate root, and define $\Delta_j = \delta_j - \lambda$, $\eta_j = \frac{\zeta_j^2}{(\delta_j - \lambda_{i-1})(\delta_j - \lambda)}$, $j = 1, 2, \ldots, n$, and

$$\phi(\tau) = \sum_{1 \le j \le n, j \ne i} \frac{\eta_j}{\Delta_j}.$$

Then the goal is to solve

$$-\frac{\eta_i}{\Delta_i - \tau} = \phi(\tau)$$

for τ.

The method is based upon approximating $\phi(\tau)$ in a neighborhood of 0 by

$$\hat{\phi}(\tau) = \frac{\Delta^2 \phi'(0)}{\Delta - \tau} + \phi(0) - \Delta\phi'(0) \approx \phi(\tau).$$

Note that the interpolation conditions

$$\phi(0) = \hat{\phi}(0), \quad \phi'(0) = \hat{\phi}'(0)$$

hold and that $\hat{\phi}(\tau)$ has a pole at Δ. We take

$$\Delta = \begin{cases} \Delta_{i+1}, & \text{if } \phi'(0) > 0; \\ \Delta_{i-1}, & \text{if } \phi'(0) \le 0. \end{cases}$$

Then the approximation to the solution τ^* is computed by solving

$$\frac{\eta_i}{\Delta_i - \tau} + \frac{\Delta^2 \phi'(0)}{\Delta - \tau} + \phi(0) - \Delta\phi'(0) = 0. \tag{6.4}$$

This equation may of course be solved via computing the solution of a quadratic equation.

It can be shown that the resulting iteration converges globally and quadratically. To demonstrate quadratic convergence we note that

$$\begin{aligned} 0 &= \frac{\eta_i}{\Delta_i - \tau^*} + \phi(\tau^*) \\ &= \frac{\eta_i(\tau^* - \tau)}{(\Delta_i - \tau)(\Delta_i - \tau^*)} + \frac{\eta_i}{\Delta_i - \tau} + \phi(\tau^*) \\ &= \frac{\eta_i(\tau^* - \tau)}{(\Delta_i - \tau^*)(\Delta_i - \tau^*)} - (\frac{\tau\Delta\phi'(0)}{\Delta - \tau} + \phi(0)) + \phi(0) + \phi'(0)\tau^* + \phi''(\vartheta)\tau^{*2}), \end{aligned}$$

and thus

$$\frac{\eta_i(\tau - \tau^*)}{(\Delta_i - \tau)(\Delta_i - \tau^*)} + \phi'(0)\frac{\Delta + \tau^*}{\Delta - \tau}(\tau - \tau^*) = -\frac{\phi'(0)\tau^{*2}}{\Delta_i - \tau} + \phi''(\vartheta)\tau^{*2}$$

so that

$$(-\frac{\phi(\tau^*)}{(\Delta_i - \tau)} + \phi'(0)\frac{\Delta + \tau^*}{\Delta - \tau})(\tau - \tau^*) = O(\tau^{*2}).$$

As long as

$$-\frac{\phi(\tau^*)}{(\Delta_i{}^*)} + \phi'(\tau^*)$$

is bounded away from zero, the expression above will imply that

$$\tau_+^* = O(\tau^{*2})$$

where $\tau^* = \lambda^* - \lambda$ and $\tau_+^* = \lambda_+^* - \lambda$.

Providing initial approximations that give upper and lower bounds on the root $\tau^* \in (\Delta_i, \Delta_{i+1})$ is a simple but important matter. Note that

$$f(\lambda) = \frac{\zeta_i^2}{\delta_i - \lambda} + \frac{\zeta_{i+1}^2}{\delta_{i+1} - \lambda} + \psi(\lambda),$$

where

$$\psi(\lambda) = 1 + \sum_{j \neq i, i+1} \frac{\zeta_j^2}{\delta_j - \lambda}.$$

Now, $\psi'(\lambda) > 0$ for $\lambda \in (\delta_i, \delta_{i+1})$, and thus

$$\psi(\delta_i) < \psi(\lambda) < \psi(\delta_{i+1}).$$

If $f(\lambda_i) = 0$, then

$$-\psi(\delta_{i+1}) < \frac{\zeta_i^2}{\delta_i - \lambda_i} + \frac{\zeta_{i+1}^2}{\delta_{i+1} - \lambda_i} < -\psi(\delta_i).$$

Solving

$$\frac{\zeta_i^2}{\delta_i - \lambda} + \frac{\zeta_{i+1}^2}{\delta_{i+1} - \lambda} = -\psi(\delta_j)$$

for the root in (δ_i, δ_{i+1}) gives a lower bound λ^l when $j = i+1$ and an upper bound λ^u when $j = i$ so that

$$\delta_i < \lambda^l < \lambda_i < \lambda^u < \delta_{i+1}.$$

These bounds have been used in the original code developed by Sorensen to implement the algorithm developed in [1].

These approximations can be used to construct an iterative algorithm to find the zeros of g.

Algorithm 5 *Given the root $\lambda_{i-1} \in (\delta_{i-1}, \delta_i)$, this algorithm locates the i-th eigenpair.*

1. *Construct an initial guess $\lambda \in (\delta_i, \delta_{i+1})$ of the root λ_i.*
2. *Put $\Delta_j = \delta_j - \lambda$ and $\eta_j = \frac{\zeta_j^2}{\delta_j - \lambda_{i-1}}$ for $j = 1, 2, \ldots, n$.*
3. *Repeat until "converge":*
   - *Solve Equation 6.4 for τ;*
   - *$\Delta_j := \Delta_j - \tau$ for $j = 1, 2, \ldots, n$; and*
   - *$\lambda := \lambda + \tau$.*

As will be shown in the following section, this iteration is successful.

7 Numerical Results

In this section, we present various numerical results obtained from the three different schemes:

WP_Sec: The original scheme in [1] and [4] which solves the secular equation solely in working precision.

Acc_Sec: The original scheme WP_Sec with the extra-precise function evalution DP_Sec invoked once in the search of each eigenpair as described in Section 5.

Reform: The scheme presented in Section 6 which solves for the zero of the inner product.

For a given $A = D + \rho z z^T$ where

$$\{\lambda_1, \lambda_2, \ldots, \lambda_n\} \qquad \text{and} \qquad Q = [q_1, \ldots, q_n]$$

is the computed eigensystem, we obtained the two common accuracy measures:

$$\max_{1 \le i \le n} \|Q^T q_i - e_i\| \,/\, n\|A\|\varepsilon \qquad \text{and} \qquad \max_{1 \le i \le n} \|Aq_i - \lambda_i q_i\| \,/\, n\varepsilon,$$

where ε is the unit roundoff and $\|\cdot\|$ is the Euclidean norm. In addition, we also tabulated the quantity

$$M := \max_i \frac{|\text{abs. error in computed } g(\lambda_i)|}{\varepsilon \cdot \min_{1 \le j \le n}\{|\Delta_j^*|\}|g'(\lambda_i)|}$$

to illustrate our analysis in Section 4.

7.1 Testing WP_Sec and Acc_Sec

We apply WP_Sec and Acc_Sec to three sets of test problems.

Test 1: These problems arise in applying the divide-and-conquer algorithm in [4] to the matrix

$$T = \begin{pmatrix} W_{21} & \beta & & & & \\ \beta & W_{21} & \beta & & & \\ & \beta & W_{21} & & & \\ & & & \ddots & & \\ & & & & \beta & \\ & & & & \beta & W_{21} \end{pmatrix},$$

where W_{21} is the symmetric tridiagonal matrix of order 21 with diagonal elements $(10, 9, \ldots, 1, 0, 1, 2, \ldots, 10)$ and off-diagonal elements all 1's. The value of β is chosen to be small. The matrix W_{21} is an example devised by Wilkinson to illustrate difficulties that algorithms might have with nearly equal eigenvalues. The matrix has pairs of extremely close eigenvalues. The matrix T can be made arbitrarily large of order $k \times 21$ by adjoining k copies of W_{21}. The resulting matrix will have eigenvalues in k clusters, eventually giving rise to difficult rank-1 update problems.

Test 2: This is a simple four-dimensional problem that illustrates the need for extra-precise function evaluation. Set $n := 4$, $\delta_1 = 1$, $\delta_4 = 3\frac{1}{3}$, $\delta_2 = 2 - \beta$, and $\delta_3 = 2 + \beta$, where $\beta = 10^{-l}$ is a parameter. By specifying $w^T = [2, \beta, \beta, 2]$, $\rho = \|w\|^2$, and $z = w/\|w\|$, we make $\lambda_2 = 2$ an eigenvalue. Thus,

$$\frac{\max \left| \rho \frac{\zeta_j^2}{\delta_j - \lambda_2} \right|}{|\delta_2 - \lambda_2| \sum_{j=1}^{n} \rho \frac{\zeta_j^2}{(\delta_j - \lambda_2)^2}} \geq \frac{1}{3\beta}.$$

That z, ρ, and δ_j's are not exactly representable in computer arithmetic does not affect the nature of this test.

Test 3: Here $n = 10$, $\delta_1 = 1$, $\delta_{10} = 3$, the rest of the δ's are of the form $2 \pm j \cdot \beta$, $j = 1, 2, 3, 4$ and $\beta = 10^{-l}$ for various values of l. Random vectors w whose 10 components satisfy $\max |w_l / w_m| \leq 2$ are generated. Then

$$\rho := \|w\|^2 \qquad \text{and} \qquad z := w/\|w\|.$$

This problem illustrates that as long as $\max_{l,m} |\zeta_l / \zeta_m|$ is moderate, working precision suffices even in the case of clustering eigenvalues.

Table 1 summarizes the test results. Note that the parameters β are chosen so that the problems are not to be deflated.

We make several observations on Table 1. First, our analysis in Section 4 is confirmed. Whenever $M \approx 10^k$ for some $k > 0$, roughly k digits are lost in orthogonality when using

Table 1: Testing WP_Sec and Acc_Sec
Machine: Sun 4; Precision: IEEE Double

Test	n	$-\log_{10}(\beta)$	$\log_{10}(M)$	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\epsilon$		$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\epsilon\|\|A\|\|$	
				WP_Sec	Acc_Sec	WP_Sec	Acc_Sec
1	20	7	5.88	1.1×10^{4}	1.4×10^{-1}	4.2×10^{-2}	8.0×10^{-2}
	20	7	8.55	1.9×10^{6}	1.6×10^{-1}	3.8×10^{-2}	5.1×10^{-2}
	20	7	4.30	3.2×10^{2}	1.0×10^{-1}	3.8×10^{-2}	6.8×10^{-2}
	48	7	9.40	2.9×10^{7}	8.5×10^{-2}	1.8×10^{-2}	2.8×10^{-2}
	36	7	11.95	8.6×10^{13}	1.1×10^{-1}	5.4×10^{1}	5.1×10^{-2}
2	4	1	0.76	1.0	5.1×10^{-1}	3.5×10^{-2}	5.5×10^{-2}
		4	3.72	1.7×10^{3}	4.1×10^{-1}	2.2×10^{-1}	1.2×10^{-1}
		7	6.72	1.4×10^{5}	5.1×10^{-1}	8.9×10^{-2}	5.2×10^{-2}
		10	9.72	8.2×10^{8}	2.9×10^{-1}	2.0×10^{-1}	2.0×10^{-1}
		13	12.72	4.0×10^{14}	3.9×10^{-1}	3.1×10^{1}	2.1×10^{-1}
3	10	1	0.06	2.1×10^{-1}	2.5×10^{-1}	6.2×10^{-2}	1.1×10^{-1}
		4	0.08	2.2×10^{-1}	1.5×10^{-1}	6.9×10^{-2}	5.1×10^{-2}
		7	0.08	2.0×10^{-1}	1.2×10^{-1}	4.1×10^{-2}	9.9×10^{-2}
		10	0.08	1.3×10^{-1}	1.6×10^{-1}	6.9×10^{-2}	7.5×10^{-2}
		13	0.10	1.1×10^{-1}	1.5×10^{-1}	6.9×10^{-2}	5.0×10^{-2}

WP_Sec, a phenomenon suggesting that some of the calculated differences $\delta_j - \lambda_i$ are losing k digits. Second, Acc_Sec works well (as expected). That full orthogonality is maintained even in cases where $M \approx 10^{13}$ illustrates that indeed our simulation is capable of evaluating the function as if "doubled-precision" were used in all intermediate calculations. Finally, the fact that WP_Sec is able to produce small $\|Aq - \lambda q\|$ is not surprising. A simple analysis similar to the one in Section 4 shows that

$$\|Aq - \lambda q\| \leq M \cdot \frac{|\text{abs. error in } f|}{\sqrt{\sum \frac{\zeta_j^2}{\Delta_j^2}}},$$

where M is a moderate constant. Thus, when f is evaluated in working precision alone,

$$\begin{aligned} \|Aq - \lambda q\| &\leq M \cdot \frac{\varepsilon \cdot \max_j \left|\frac{\zeta_j^2}{\Delta_j}\right|}{\sqrt{\sum \frac{\zeta_j^2}{\Delta_j^2}}} \\ &\leq M\varepsilon \cdot \max_j |\zeta_j| \\ &\leq M\varepsilon. \end{aligned}$$

Next, we illustrate the portability of our simulation by running our Fortran code on a VAX 8700 (under VMS) and an IBM 370. The precisions we used are Real*8 and Real*16. Although Real*8 is not the highest precision in those environments, we still use our simulation technique instead of a simple computation using Real*16. In the Real*8 case, the Fortran code that ran on the Sun needed no change to run correctly on the VAX and the IBM. In the Real*16 case, the only change required was to alter all declarations from Real*4 and Real*8 to Real*8 and Real*16, respectively. Table 2 summarizes the results. Residues 1 and 2 are the measures

$$\max_i \|Q^T q_i - e_i\| \Big/ n\varepsilon \qquad \text{and} \qquad \max_i \|Aq_i - \lambda_i q_i\| \Big/ n\varepsilon\|A\|,$$

respectively.

7.2 Testing Reform

The reformulation solves the i-th eigenpair based on the $(i-1)$-th eigenvalue. Thus, the obvious implementation uses WP_Sec to locate the first eigenpair and then uses the reformulation to locate the remaining eigenpairs $(\lambda_2, q_2), (\lambda_3, q_3), \ldots, (\lambda_n, q_n)$ in order. Recall that the reformulation may reduce the difficulties caused by rounding error in evaluating the secular equation. The measure of difficulty is quantified by our measure

$$M = \frac{|\text{abs. error in computed function}|}{|\text{root}| \cdot |\text{derivative of function at root}|}.$$

Tables 3 and 4 compare WP_Sec with Reform on our first two sets of tests. Note the improvements in M and the residues as a result of using Reform.

Table 2: Testing Acc_Sec on Various Machines

Machine	Precision	$-\log_2(\varepsilon)$	Residue 1 for Test 1	Residue 1 for Test 2	Residue 1 for Test 3	Residue 2 for Test 1	Residue 2 for Test 2	Residue 2 for Test 3
Sun 4	IEEE Double	52	0.16	0.51	0.25	0.08	0.21	0.11
VAX/VMS	D-format	55	0.22	0.14	0.61	0.10	0.11	0.22
	G-format	52	0.21	0.13	0.63	0.11	0.08	0.21
	H-format	112	0.36	0.10	0.53	0.11	0.08	0.22
IBM 370	IBM Double	52	0.41	0.37	0.97	0.15	0.07	0.77
	IBM Quad	108	0.42	0.25	0.61	0.16	0.08	0.42

Table 3: Comparing WP_Sec and Reform on Test 1
Machine: Sun 4; Precision: IEEE Double

n	$\log_{10}(M)$ WP_Sec	$\log_{10}(M)$ Reform	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$ WP_Sec	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$ Reform	$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$ WP_Sec	$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$ Reform
20	5.88	0.04	1.1×10^{4}	1.0×10^{-1}	4.2×10^{-2}	4.2×10^{-2}
20	8.55	0.08	1.9×10^{6}	1.6×10^{-1}	3.8×10^{-2}	6.0×10^{-2}
20	4.30	0.04	3.2×10^{2}	1.3×10^{-1}	3.8×10^{-2}	5.3×10^{-2}
48	9.40	0.77	2.9×10^{7}	8.7×10^{-2}	1.8×10^{-2}	2.2×10^{-2}
36	11.95	1.08	8.6×10^{13}	1.1×10^{-1}	5.4×10^{1}	4.0×10^{-2}

Table 4: Comparing WP_Sec and Reform on Test 2
Machine: Sun 4; Precision: IEEE Double

$-\log_2(\beta)$	$\log_{10}(M)$ WP_Sec	$\log_{10}(M)$ Reform	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$ WP_Sec	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$ Reform	$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$ WP_Sec	$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$ Reform
1	0.76	0.75	1.0	3.8×10^{-1}	3.5×10^{-2}	4.9×10^{-1}
4	3.72	0.60	1.7×10^{3}	3.4	2.2×10^{-1}	1.3
7	6.72	0.61	1.4×10^{5}	2.0×10^{-1}	8.9×10^{-2}	1.0
10	9.72	0.60	8.2×10^{8}	2.6×10^{-1}	2.0×10^{-1}	1.5
13	12.72	0.68	4.0×10^{14}	3.8×10^{-1}	3.1×10^{1}	4.9×10^{-1}

Table 5: Comparing WP_Sec and Reform on Test 3
Machine: Sun 4; Precision: IEEE Double

	$\log_{10}(M)$		$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$		$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$	
$-\log_{10}(\beta)$	WP_Sec	Reform	WP_Sec	Reform	WP_Sec	Reform
1	0.06	0.83	2.1×10^{-1}	1.5×10^{-1}	6.2×10^{-2}	1.0×10^{-1}
4	0.08	2.69	2.2×10^{-1}	2.0×10^{2}	6.9×10^{-2}	3.4×10^{3}
7	0.08	5.68	2.0×10^{-1}	1.9×10^{5}	4.1×10^{-2}	3.6×10^{6}
10	0.08	8.79	1.3×10^{-1}	3.0×10^{8}	6.9×10^{-2}	4.4×10^{9}
13	0.10	11.7	1.1×10^{-1}	3.3×10^{10}	6.9×10^{-2}	4.6×10^{11}

Is Reform better than WP_Sec? The most obvious drawback is that Reform is intrinsically serial. A good parallel implementation of the divide-and-conquer scheme in [4] using Reform would have to divide the δ's into several groups and apply the method to each group in parallel; i.e. the first eigenpair of each group is solved by WP_Sec before the remaining eigenpairs are located using Reform.

But Reform has another problem. It can fail miserably while WP_Sec succeeds. Table 5 illustrates the situation. The values of the M's tell us that reformulating these problems changes them from easy to hard — a scientific illustration that reformations are not all beneficial.

In short, Reform must be used cautiously. WP_Sec should be used as long as M is moderate. A switch to Reform should be made only when WP_Sec yields a large M. But even then, this combination of WP_Sec and Reform offers no guarantee of high accuracy for all possible problems. Clearly, Acc_Sec is our choice!

8 Comparison with Kahan's Method

We first reported in the summer of 1988 [12] our experimental success with the two approaches here. A more recent presentation can be found in the 1989 Numerical Analysis conference at Dundee [11]. The full analysis is more recent work. Kahan studied the same problem independently and documented his work in an unpublished manuscript "Rank-1 Perturbed Diagonal's Eigensystem" dated July 1989. In this section, we summarize Kahan's method and offer some comparisons with ours.

8.1 Kahan's Method

The gist of Kahan's method is similar to our scheme Acc_Sec. To compute the i-th eigenpair, Kahan considered the translated secular equation

$$g(x) \;=\; \frac{1}{\rho} x f(\delta_I + x)$$

$$= x\left(\frac{1}{\rho} + \sum_{j\neq I} \frac{\zeta_j^2}{(\delta_j - \delta_I) - x} - \frac{\zeta_I^2}{x}\right)$$

$$= x\gamma_I + x^2 \sum_{j\neq I} \frac{\zeta_j^2}{(\delta_j - \delta_I)(\delta_j - \delta_I - x)} - \zeta_I^2,$$

where

$$\gamma_I = \frac{1}{\rho} + \sum_{j\neq I} \frac{\zeta_j^2}{\delta_j - \delta_I}.$$

Clearly, the root x^* of $g(x)$ that Kahan sought is the Δ_I^* that we tried to locate. Next, Kahan computed γ_I using simulated extended precision before rounding the value back to working precision. Note that γ_I's form is similar to the function we tackled in Acc_Sec. The simulation technique is similar to ours. (This is not surprising since the second author learned those techniques from Kahan in the first place.) After the working-precision γ_I was obtained, Kahan solved $g(x) = 0$ using solely working precision.

8.2 Kahan's Analysis

Kahan offered only a brief analysis of his proposed method. The conclusion of his analysis was that in order to compute the root accurately enough to ensure orthogonality, the rounding errors in the computation of γ_I are potentially harmful and therefore have to be "attenuated by the use of doubled-precision arithmetic." Is such doubled precision really sufficient in his case?

8.3 Examples and Counterexamples

Kahan coded his method in Basic on an IBM-XT with an Intel 8087 coprocessor. We first ran the program "TestAbp" (the executable module that he supplied us) on such a machine with the difficult subproblems from the W_{21} test described previously. The eigensystems were all calculated to full machine precision in those cases. Clearly, in those difficult cases where our analysis showed that a "doubled-precision" evaluation of f is necessary, the constant γ_I must have accounted for most of the cancellation.

Unfortunately, situations are not always that favorable. Let us apply our analysis in Section 4 here. Since

$$g(x) = x\gamma_I + x^2 \sum_{j\neq I} \frac{\zeta_j^2}{(\delta_j - \delta_I)(\delta_j - \delta_I - x)} - \zeta_I^2$$

is evaluated in working precision,

$$|\text{abs. error in } g| \geq \varepsilon|x\gamma_I|$$

Table 6: Kahan's TestAbp on 2 × 2 Examples

$-\log_{10}(\beta)$	$\max_i \|Q^T q_i - e_i\| \,/\, n\varepsilon$	$\max_i \|Aq_i - \lambda_i q_i\| \,/\, n\varepsilon\|A\|$
4	4.1×10^{-1}	8.9×10^{2}
5	6.4×10^{-1}	5.7×10^{3}
6	4.5×10^{-1}	2.8×10^{5}
7	8.2×10^{-2}	3.6×10^{5}
8	5.3×10^{-1}	5.0×10^{6}

in general. Since $g(x) = \frac{x}{\rho} f(\delta_I + x)$, at $x^* = \lambda_i - \delta_I$,

$$\begin{aligned} |g'(x^*)| &= \left| \frac{x^*}{\rho} f'(\lambda_i) \right| \\ &= \left| \frac{x^*}{\rho} \right| \sum_{j=1}^{n} \frac{\zeta_j^2}{(\delta_j - \lambda_i)^2}. \end{aligned}$$

The indicator of accuracy is

$$\begin{aligned} M &= |\text{abs. error in } g| \,/\, \varepsilon |x^* g'(x^*)| \\ &= |\rho \gamma_I| \,/\, \sum_{j=1}^{n} \frac{\zeta_j^2}{(\delta_j - \lambda_i)^2}. \end{aligned}$$

Hence, if $|\gamma_I|$ is large compared to $\sum \frac{\zeta_j^2}{(\delta_j - \lambda_i)^2}$, accuracy will be lost. The following 2×2 example illustrates the situation.

Define $\delta_1 := 1 - \beta$ $(0 < \beta < 1)$, $\delta_2 := 1$, $w^T := [w_1, w_2]$ where

$$w_1 := \sqrt{(1+\beta)/2}, \quad w_2 := \sqrt{1/2}, \quad \rho := \|w\|^2, \quad \text{and} \quad z := w/\|w\|.$$

It is easy to show that $\lambda_2 = 2$ is an eigenvalue. But at $i = 2$, $\gamma \approx 1/\beta$ and $f'(\lambda_2) \approx 1$. Hence by choosing $\beta = 10^{-l}$, $l > 0$, we expect an error of 10^l in either $\|Q^T q_i - e_i\|/\|A\|\varepsilon$ or $\|Aq_i - \lambda_i q_i\|/n\varepsilon$, or both. Table 6 confirms our analysis. That the error grows like 10^{l-2} instead of 10^l is probably due to the fact that the Intel 8087 coprocessor does all arithmetic to roughly 3 digits more than IEEE double precision.

In the same way, we created some 4×4 examples as follows: $n := 4$, $\delta_1 := 1 - 4\beta$, $\delta_2 := 1$, $\delta_3 := 2 + 3\beta$, $\delta_4 := 2 + 4\beta$,

$$w^T := [1, 1, \frac{1}{2} + 3\beta, 1], \quad \rho := \|w\|^2, \quad \text{and} \quad z := w/\|w\|.$$

Once again, errors of the order of $1/\beta$ are exhibited, as shown in Table 7. Ironically, the simple method WP_Sec (without simulation) works fine on both sets of problems. (This is expected, since all the components of z are of comparable magnitude.)

Table 7: Kahan's TestAbp on 4 × 4 Examples

$-\log_{10}(\beta)$	$\max_i \|\|Q^T q_i - e_i\|\| \,/\, n\varepsilon$	$\max_i \|\|Aq_i - \lambda_i q_i\|\| \,/\, n\varepsilon\|\|A\|\|$
4	2.3×10^2	5.0×10^2
5	6.5×10^2	4.7×10^3
6	4.7×10^3	2.8×10^4
7	1.8×10^5	8.3×10^4
8	1.1×10^6	2.9×10^6

Towards the end of his presentation of his method, Kahan mentioned that an alternative would be to search for the root of

$$\tilde{g}(x) = \frac{1}{\rho} x f(\vartheta + x),$$

where ϑ is a better approximation to λ_i than δ_I is. The associated constant γ_ϑ would then be calculated by using extra-precision arithmetic. Kahan stated that this "alternative has not been found necessary, so it has not been elaborated, though it may deserve to be explored."

But the alternative *is* necessary. In fact, our scheme here is exactly the extreme end of the alternative: the ϑ is simply the root of f in (δ_i, δ_{i+1}) computed by using working precision alone. At this point, the magnitude of f is approximately the absolute error in evaluating f. Acc_Sec is then invoked, which is tantamount to calculating γ_ϑ. Had Kahan explored and implemented this alternative, the differences between his and our methods would have been only the root finder scheme. Kahan used quadratic interpolation while we use rational interpolation. We believe that a robust and efficient scheme can very likely benefit from both interpolations.

9 Concluding Remarks

We have shown that the important problem of determining the eigensystems of rank-1 perturbed diagonal matrices can be solved very accurately. Moreover, good computer arithmetic, most notably IEEE 754, has made highly reliable software for this problem possible. Good arithmetic hardware, however, is not the only necessary ingredient for such software. Language must offer good support to the hardware environment at hand; and compilers must appreciate various subtle uses for floating point; for example, the compilers must at least be controllable not to take various "optimization techniques" such as rearranging the order of computations and thus destroying the accuracy of carefully crafted codes.

Finally, it is unfortunate that neither Reform nor Acc_Sec would work on machines whose arithmetic subtraction lacks a guard digit (or bit); notable examples are Crays

and CDCs. It is even more unfortunate because, despite the many anomolies those arithmetics offer, merely adding a guard digit (or bit) would allow Acc_Sec to work. The ultimate misfortune is that robust programs such as Acc_Sec may never appear in a federally funded software library such as LAPACK simply because the program may not run on a few aberrant machines whose manufacturers have not implemented the extra guard digit in thier hardware. These include some of the most powerful high performance computers.

References

[1] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, Rank-one modification of the symmetric eigenproblem, *Numerische Mathematik*, 31, pp. 31-48, 1978.

[2] J. J. M. Cuppen, A divide and conquer method for the symmetric tridiagonal eigenproblem, *Numerische Mathematik*, 36, pp. 177-195, 1981.

[3] T. J. Dekker, A floating-point technique for extending the available precision, *Numerische Mathematik*, 18, pp. 224-242, 1971.

[4] J. J. Dongarra and D. C. Sorensen, A fully parallel algorithm for the symmetric eigenproblem, *SIAM J. Sci. Stat. Comput.*, 8, pp. s139-s154, 1987.

[5] J. J. Dongarra and D. C. Sorensen, On the implementation of a fully parallel algorithm for the symmetric eigenproblem, SPIE Vol. 696, *Advanced Algorithms and Architectures for Signal Processing*, pp. 45-53, 1986.

[6] J. E. Dennis and J. J. Moré, A characterization of superlinear convergence and its application to quasi-Newton methods, *Mathematics of Computation*, 28, pp. 549-560, 1974.

[7] G. H. Golub, Some modified matrix eigenvalue problems, *SIAM Review*, 15, pp. 318-334, 1973.

[8] IEEE Standard for Binary Floating-Point Arithmetic, *ANSI/IEEE Standard 754-1985*, Institute of Electrical and Electronic Engineers, New York, 1985.

[9] I. C. F. Ipsen and E. R. Jessup, Solving the symmetric tridiagonal eigenvalue problem on the Hypercube, *SIAM J. Sci. Stat. Comput.*, 11, pp. 203-229, 1990.

[10] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, Matrix Eigensystem Routines - EISPACK Guide, *Lecture Notes in Computer Science*, Vol. 6, 2nd edition, Springer-Verlag, Berlin, 1976.

[11] D. C. Sorensen, Numerical aspects of divide and conquer methods for eigenvalue problems, in *Numerical Analysis: Volume III,* edited by D. F. Griffiths and G. A. Watson, Longman, London, 1989.

[12] D. C. Sorensen, On the orthogonality of eigenvectors computed by a divide and conquer technique, *Proceedings of the 1988 IBM European Institute Seminar on Parallel Computing,* Oberlech, Austria, July 1988.

[13] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.