Chapter 4

Cuppen's Divide and Conquer Algorithm

In this chapter we deal with an algorithm that is designed for the efficient solution of the symmetric tridiagonal eigenvalue problem

$$T\mathbf{x} = \lambda\mathbf{x}, \qquad T = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & b_{n-1} \\ & & b_{n-1} & a_n \end{bmatrix}. \tag{4.1}$$

We noticed from Table 3.1 that the reduction of a full symmetric matrix to a similar tridiagonal matrix requires about $\frac{8}{3}n^3$ while the tridiagonal QR algorithm needs an estimated $6n^3$ floating operations (flops) to converge. Because of the importance of this subproblem a considerable effort has been put into finding faster algorithms than the QR algorithms to solve the tridiagonal eigenvalue problem. In the mid-1980's Dongarra and Sorensen [4] promoted an algorithm originally proposed by Cuppen [2]. This algorithm was based on a divide and conquer strategy. However, it took ten more years until a stable variant was found by Gu and Eisenstat [5, 6]. Today, a stable implementation of this latter algorithm is available in LAPACK [1].

4.1 The divide and conquer idea

Divide and conquer is an old strategy in military to defeat an enemy going back at least to Caesar. In computer science, divide and conquer (D&C) is an important algorithm design paradigm. It works by recursively breaking down a problem into two or more subproblems of the same (or related) type, until these become simple enough to be solved directly. The solutions to the subproblems are then combined to give a solution to the original problem. Translated to our problem the strategy becomes

1. Partition the tridiagonal eigenvalue problem into two (or more) smaller tridiagonal eigenvalue problems.
2. Solve the two smaller problems.
3. Combine the solutions of the smaller problems to get the desired solution of the overall problem.

Evidently, this strategy can be applied recursively.

4.2 Partitioning the tridiagonal matrix

Partitioning the *irreducible* tridiagonal matrix is done in the following way. We write

(4.2)

$$
\begin{aligned}
T &= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & a_2 & \ddots & & & & & & & \\
 & \ddots & \ddots & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m & b_m & & & & & \\
\hline
 & & & b_m & a_{m+1} & b_{m+1} & & & & \\
 & & & & b_{m+1} & a_{m+2} & \ddots & & & \\
 & & & & & \ddots & \ddots & b_{n-1} & & \\
 & & & & & & b_{n-1} & a_n &
\end{array}\right] \\
&= \left[\begin{array}{ccccc|ccccc}
a_1 & b_1 & & & & & & & & \\
b_1 & a_2 & \ddots & & & & & & & \\
 & \ddots & \ddots & b_{m-1} & & & & & & \\
 & & b_{m-1} & a_m \mp b_m & & & & & & \\
\hline
 & & & & a_{m+1} \mp b_m & b_{m+1} & & & & \\
 & & & & b_{m+1} & a_{m+2} & \ddots & & & \\
 & & & & & \ddots & \ddots & b_{n-1} & & \\
 & & & & & & b_{n-1} & a_n &
\end{array}\right] + \left[\begin{array}{ccccc|ccccc}
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & \pm b_m & b_m & & & & \\
\hline
 & & & & b_m & \pm b_m & & & & \\
 & & & & & & & & & \\
 & & & & & & & & & \\
 & & & & & & & & &
\end{array}\right] \\
&= \left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho \mathbf{u}\mathbf{u}^T \qquad \text{with } \mathbf{u} = \left[\begin{array}{c} \pm\,\mathbf{e}_m \\ \hline \mathbf{e}_1 \end{array}\right] \text{ and } \rho = \pm\, b_m,
\end{aligned}
$$

where $\mathbf{e}_m$ is a vector of length $m \approx \frac{n}{2}$ and $\mathbf{e}_1$ is a vector of length $n - m$. Notice that the most straightforward way to partition the problem without modifying the diagonal elements leads to a rank-two modification. With the approach of (4.2) we have the original T as a sum of two smaller tridiagonal systems plus a *rank-one modification*.

4.3 Solving the small systems

We solve the half-sized eigenvalue problems,

$$T_i = Q_i \Lambda_i Q_i^T, \qquad Q_i^T Q_i = I, \qquad i = 1, 2. \tag{4.3}$$

These two spectral decompositions can be computed by any algorithm, in particular also by this divide and conquer algorithm by which the T_i would be further split. It is clear that by this partitioning an large number of small problems can be generated that can be potentially solved in parallel. For a parallel algorithm, however, the further phases of the algorithm must be parallelizable as well.

Plugging (4.3) into (4.2) gives

$$\left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right] \left(\left[\begin{array}{c|c} T_1 & \\ \hline & T_2 \end{array}\right] + \rho \mathbf{u}\mathbf{u}^T \right) \left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] = \left[\begin{array}{c|c} \Lambda_1 & \\ \hline & \Lambda_2 \end{array}\right] + \rho \mathbf{v}\mathbf{v}^T \tag{4.4}$$

with

$$\mathbf{v} = \left[\begin{array}{c|c} Q_1^T & \\ & Q_2^T \end{array}\right] \mathbf{u} = \left[\begin{array}{c} \pm\, Q_1^T \mathbf{e}_m \\ Q_2^T \mathbf{e}_1 \end{array}\right] = \left[\begin{array}{c} \pm \text{ last row of } Q_1 \\ \text{first row of } Q_2 \end{array}\right]. \tag{4.5}$$

Now we have arrived at the eigenvalue problem

$$(D+\rho\mathbf{v}\mathbf{v}^T)\mathbf{x}=\lambda\mathbf{x},\qquad D=\Lambda_1\oplus\Lambda_2=\operatorname{diag}(\lambda_1,\ldots,\lambda_n). \tag{4.6}$$

That is, we have to compute the spectral decomposition of a matrix that is a **diagonal plus a rank-one update**. Let

$$D+\rho\mathbf{v}\mathbf{v}^T=Q\Lambda Q^T \tag{4.7}$$

be this spectral decomposition. Then, the spectral decomposition of the tridiagonal T is

$$T=\left[\begin{array}{c|c} Q_1 & \\ \hline & Q_2 \end{array}\right] Q\Lambda Q^T \left[\begin{array}{c|c} Q_1^T & \\ \hline & Q_2^T \end{array}\right]. \tag{4.8}$$

Forming the product $(Q_1\oplus Q_2)Q$ will turn out to be *the most expensive step of the algorithm*. It costs $n^3+\mathcal{O}(n^2)$ floating point operations

4.4 Deflation

There are certain solutions of (4.7) that can be given immediately, by just looking carefully at the equation.

If there are zero entries in $\mathbf{v}$ then we have

$$\big(v_i=0\Leftrightarrow\mathbf{v}^T\mathbf{e}_i=0\big)\quad\Longrightarrow\quad(D+\rho\mathbf{v}\mathbf{v}^T)\mathbf{e}_i=d_i\mathbf{e}_i. \tag{4.9}$$

Thus, if an entry of $\mathbf{v}$ vanishes we can read the eigenvalue from the diagonal of D at once and the corresponding eigenvector is a coordinate vector.

If identical entries occur in the diagonal of D, say $d_i=d_j$, with $i<j$, then we can find a plane rotation $G(i,j,\phi)$ (see (3.4)) such that it introduces a zero into the j-th position of $\mathbf{v}$,

$$G^T\mathbf{v}=G(i,j,\varphi)^T\mathbf{v}=\begin{bmatrix}\times\\ \vdots\\ \sqrt{{v_i}^2+{v_j}^2}\\ \vdots\\ 0\\ \vdots\\ \times\end{bmatrix}\begin{array}{l}\\ \\ \leftarrow\ i\\ \\ \leftarrow\ j\\ \\ \\ \end{array}$$

Notice, that (for *any* φ),

$$G(i,j,\varphi)^T DG(i,j,\varphi)=D,\qquad d_i=d_j.$$

So, if there are multiple eigenvalues in D we can reduce all but one of them by introducing zeros in $\mathbf{v}$ and then proceed as previously in (4.9).

When working with floating point numbers we deflate if

$$|v_i|<C\varepsilon\|T\|\qquad\text{or}\qquad|d_i-d_j|<C\varepsilon\|T\|,\qquad(\|T\|=\|D+\rho\mathbf{v}\mathbf{v}^T\|) \tag{4.10}$$

where C is a small constant. Deflation changes the eigenvalue problem for $D+\rho\mathbf{v}\mathbf{v}^T$ into the eigenvalue problem for

$$\begin{bmatrix} D_1+\rho\mathbf{v}_1\mathbf{v}_1^T & O\\ O & D_2\end{bmatrix}=G^T(D+\rho\mathbf{v}\mathbf{v}^T)G+E,\qquad\|E\|<C\varepsilon\sqrt{\|D\|^2+|\rho|^2\|\mathbf{v}\|^4}, \tag{4.11}$$

where D_1 has no multiple diagonal entries and $\mathbf{v}_1$ has no zero entries. So, we have to compute the spectral decomposition of the matrix in (4.11) which is similar to a slight perturbation of the original matrix. G is the product of Givens rotations.

4.4.1 Numerical examples

Let us first consider

$$T = \left[\begin{array}{ccc|ccc} 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 3 & 1 & & \\ \hline & & 1 & 4 & 1 & \\ & & & 1 & 5 & 1 \\ & & & & 1 & 6 \end{array}\right] = \left[\begin{array}{ccc|ccc} 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 2 & 0 & & \\ \hline & & 0 & 3 & 1 & \\ & & & 1 & 5 & 1 \\ & & & & 1 & 6 \end{array}\right] + \left[\begin{array}{ccc|ccc} 0 & & & & & \\ & 0 & & & & \\ & & 1 & 1 & & \\ \hline & & 1 & 1 & & \\ & & & & 0 & \\ & & & & & 0 \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc} 1 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 2 & & & \\ \hline & & & 3 & 1 & \\ & & & 1 & 5 & 1 \\ & & & & 1 & 6 \end{array}\right] + \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 1 \\ 0 \\ 0 \end{array}\right] \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 1 \\ 0 \\ 0 \end{array}\right]^T = T_0 + \mathbf{u}\mathbf{u}^T.$$

Then a little MATLAB experiment shows that

$$Q_0^T T Q_0 = \begin{bmatrix} 0.1981 & & & & & & \\ & 1.5550 & & & & & \\ & & 3.2470 & & & & \\ & & & 2.5395 & & & \\ & & & & 4.7609 & & \\ & & & & & 6.6996 \end{bmatrix} + \begin{bmatrix} 0.3280 \\ 0.7370 \\ 0.5910 \\ 0.9018 \\ -0.4042 \\ 0.1531 \end{bmatrix} \begin{bmatrix} 0.3280 \\ 0.7370 \\ 0.5910 \\ 0.9018 \\ -0.4042 \\ 0.1531 \end{bmatrix}^T$$

with

$$Q_0 = \begin{bmatrix} 0.7370 & -0.5910 & 0.3280 & & & \\ -0.5910 & -0.3280 & 0.7370 & & & \\ 0.3280 & 0.7370 & 0.5910 & & & \\ & & & 0.9018 & -0.4153 & 0.1200 \\ & & & -0.4042 & -0.7118 & 0.5744 \\ & & & 0.1531 & 0.5665 & 0.8097 \end{bmatrix}$$

Here it is not possible to deflate.

Let us now look at an example with more symmetry,

$$T = \left[\begin{array}{ccc|ccc} 2 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 2 & 1 & & \\ \hline & & 1 & 2 & 1 & \\ & & & 1 & 2 & 1 \\ & & & & 1 & 2 \end{array}\right] = \left[\begin{array}{ccc|ccc} 2 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 1 & 0 & & \\ \hline & & 0 & 1 & 1 & \\ & & & 1 & 2 & 1 \\ & & & & 1 & 2 \end{array}\right] + \left[\begin{array}{ccc|ccc} 0 & & & & & \\ & 0 & & & & \\ & & 1 & 1 & & \\ \hline & & 1 & 1 & & \\ & & & & 0 & \\ & & & & & 0 \end{array}\right]$$

$$= \left[\begin{array}{ccc|ccc} 2 & 1 & & & & \\ 1 & 2 & 1 & & & \\ & 1 & 1 & & & \\ \hline & & & 1 & 1 & \\ & & & 1 & 2 & 1 \\ & & & & 1 & 2 \end{array}\right] + \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 1 \\ 0 \\ 0 \end{array}\right] \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ \hline 1 \\ 0 \\ 0 \end{array}\right]^T = T_0 + \mathbf{u}\mathbf{u}^T.$$

Now, MATLAB gives

$$Q_0^T T Q_0 = \begin{bmatrix} 0.1981 & & & & & & \\ & 1.5550 & & & & & \\ & & 3.2470 & & & & \\ & & & 0.1981 & & & \\ & & & & 1.5550 & & \\ & & & & & 3.2470 \end{bmatrix} + \begin{bmatrix} 0.7370 \\ -0.5910 \\ 0.3280 \\ 0.7370 \\ -0.5910 \\ 0.3280 \end{bmatrix} \begin{bmatrix} 0.7370 \\ -0.5910 \\ 0.3280 \\ 0.7370 \\ -0.5910 \\ 0.3280 \end{bmatrix}^T$$

with

$$Q_0 = \begin{bmatrix} 0.3280 & 0.7370 & 0.5910 & & & \\ -0.5910 & -0.3280 & 0.7370 & & & \\ 0.7370 & -0.5910 & 0.3280 & & & \\ & & & 0.7370 & -0.5910 & 0.3280 \\ & & & -0.5910 & -0.3280 & 0.7370 \\ & & & 0.3280 & 0.7370 & 0.5910 \end{bmatrix}$$

In this example we have *three* double eigenvalues. Because the corresponding components of $\mathbf{v}$ (v_i and v_{i+1}) are equal we define

$$\begin{aligned} G &= G(1,4,\pi/4)G(2,5,\pi/4)G(3,6,\pi/4) \\ &= \left[\begin{array}{ccc|ccc} 0.7071 & & & 0.7071 & & \\ & 0.7071 & & & 0.7071 & \\ & & 0.7071 & & & 0.7071 \\ \hline -0.7071 & & & 0.7071 & & \\ & -0.7071 & & & 0.7071 & \\ & & -0.7071 & & & 0.7071 \end{array}\right]. \end{aligned}$$

Then,

$$\begin{aligned} G^T Q_0^T T Q_0 G &= G^T Q_0^T T_0 Q_0 G + G^T\mathbf{v}(G^T\mathbf{v})^T = D + G^T\mathbf{v}(G^T\mathbf{v})^T \\ &= \begin{bmatrix} 0.1981 & & & & & & \\ & 1.5550 & & & & & \\ & & 3.2470 & & & & \\ & & & 0.1981 & & & \\ & & & & 1.5550 & & \\ & & & & & 3.2470 \end{bmatrix} + \begin{bmatrix} 1.0422 \\ -0.8358 \\ 0.4638 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix} \begin{bmatrix} 1.0422 \\ -0.8358 \\ 0.4638 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}^T \end{aligned}$$

Therefore, (in this example) $\mathbf{e}_4$, $\mathbf{e}_5$, and $\mathbf{e}_6$ are eigenvectors of

$$D + G^T\mathbf{v}(G^T\mathbf{v})^T = D + G^T\mathbf{v}\mathbf{v}^T G$$

corresponding to the eigenvalues d_4, d_5, and d_6, respectively. The eigenvectors of T corresponding to these three eigenvalues are the last three columns of

$$Q_0 G = \begin{bmatrix} 0.2319 & -0.4179 & 0.5211 & 0.5211 & -0.4179 & 0.2319 \\ 0.5211 & -0.2319 & -0.4179 & -0.4179 & -0.2319 & 0.5211 \\ 0.4179 & 0.5211 & 0.2319 & 0.2319 & 0.5211 & 0.4179 \\ -0.2319 & 0.4179 & -0.5211 & 0.5211 & -0.4179 & 0.2319 \\ -0.5211 & 0.2319 & 0.4179 & -0.4179 & -0.2319 & 0.5211 \\ -0.4179 & -0.5211 & -0.2319 & 0.2319 & 0.5211 & 0.4179 \end{bmatrix}.$$

4.5 The eigenvalue problem for $D + \rho\mathbf{v}\mathbf{v}^T$

We know that $\rho \neq 0$. Otherwise there is nothing to be done. Furthermore, after deflation, we know that all elements of $\mathbf{v}$ are nonzero and that the diagonal elements of D are all

distinct, in fact,

$$|d_i - d_j| > C\varepsilon\|T\|.$$

We order the diagonal elements of D such that

$$d_1 < d_2 < \cdots < d_n.$$

Notice that this procedure permutes the elements of $\mathbf{v}$ as well. Let $(\lambda, \mathbf{x})$ be an eigenpair of

$$(D + \rho\mathbf{v}\mathbf{v}^T)\mathbf{x} = \lambda\mathbf{x}. \tag{4.12}$$

Then,

$$(D - \lambda I)\mathbf{x} = -\rho\mathbf{v}\mathbf{v}^T\mathbf{x}. \tag{4.13}$$

λ cannot be equal to one of the d_i. If $\lambda = d_k$ then the k-th element on the left of (4.13) vanishes. But then either $v_k = 0$ or $\mathbf{v}^T\mathbf{x} = 0$. The first cannot be true for our assumption about $\mathbf{v}$. If on the other hand $\mathbf{v}^T\mathbf{x} = 0$ then $(D - d_k I)\mathbf{x} = \mathbf{0}$. Thus $\mathbf{x} = \mathbf{e}_k$ and $\mathbf{v}^T\mathbf{e}_k = v_k = 0$, which cannot be true. Therefore $D - \lambda I$ is nonsingular and

$$\mathbf{x} = \rho(\lambda I - D)^{-1}\mathbf{v}(\mathbf{v}^T\mathbf{x}). \tag{4.14}$$

This equation shows that $\mathbf{x}$ is proportional to $(\lambda I - D)^{-1}\mathbf{v}$. If we require $\|\mathbf{x}\| = 1$ then

$$\mathbf{x} = \frac{(\lambda I - D)^{-1}\mathbf{v}}{\|(\lambda I - D)^{-1}\mathbf{v}\|}. \tag{4.15}$$

Multiplying (4.14) by $\mathbf{v}^T$ from the left we get

$$\mathbf{v}^T\mathbf{x} = \rho\mathbf{v}^T(\lambda I - D)^{-1}\mathbf{v}(\mathbf{v}^T\mathbf{x}). \tag{4.16}$$

Since $\mathbf{v}^T\mathbf{x} \neq 0$, λ is an eigenvalue of (4.12) if and only if

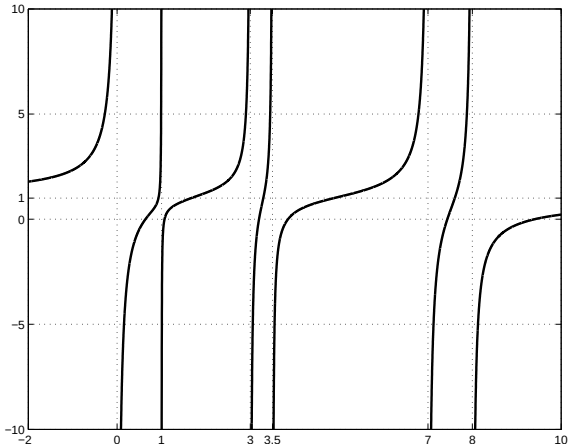

Figure 4.1: Graph of $1 + \frac{1}{0-\lambda} + \frac{0.2^2}{1-\lambda} + \frac{0.6^2}{3-\lambda} + \frac{0.5^2}{3.5-\lambda} + \frac{0.9^2}{7-\lambda} + \frac{0.8^2}{8-\lambda}$

$$\boxed{f(\lambda) := 1 - \rho\mathbf{v}^T(\lambda I - D)^{-1}\mathbf{v} = 1 - \rho\sum_{k=1}^{n}\frac{v_k^2}{\lambda - d_k} = 0.} \tag{4.17}$$

This equation is called **secular equation**. The secular equation has **poles** at the eigenvalues of D and **zeros** at the eigenvalues of $D + \rho\mathbf{v}\mathbf{v}^T$. Notice that

$$f'(\lambda) = \rho \sum_{k=1}^{n} \frac{v_k^2}{(\lambda - d_k)^2}.$$

Thus, the derivative of f is positive if $\rho > 0$ wherever it has a finite value. If $\rho < 0$ the derivative of f is negative (almost) everywhere. A typical graph of f with $\rho > 0$ is depicted in Fig. 4.1. (If ρ is negative the image can be flipped left to right.) The secular equation implies the **interlacing property** of the eigenvalues of D and of $D + \rho\mathbf{v}\mathbf{v}^T$,

$$d_1 < \lambda_1 < d_2 < \lambda_2 < \cdots < d_n < \lambda_n, \qquad \rho > 0. \tag{4.18}$$

or

$$\lambda_1 < d_1 < \lambda_2 < d_2 < \cdots < \lambda_n < d_n, \qquad \rho < 0. \tag{4.19}$$

So, we have to compute one eigenvalue in each of the intervals (d_i, d_{i+1}), $1 \le i < n$, and a further eigenvalue in (d_n, ∞) or $(-\infty, d_1)$. The corresponding eigenvector is then given by (4.15). Evidently, these tasks are easy to parallelize.

Equations (4.17) and (4.15) can also been obtained from the relations

$$\begin{aligned}
\begin{bmatrix} \frac{1}{\rho} & \mathbf{v}^T \\ \mathbf{v} & \lambda I - D \end{bmatrix} &= \begin{bmatrix} 1 & \mathbf{0}^T \\ \rho\mathbf{v} & I \end{bmatrix} \begin{bmatrix} \frac{1}{\rho} & \mathbf{0}^T \\ \mathbf{0} & \lambda I - D - \rho\mathbf{v}\mathbf{v}^T \end{bmatrix} \begin{bmatrix} 1 & \rho\mathbf{v}^T \\ \mathbf{0} & I \end{bmatrix} \\
&= \begin{bmatrix} 1 & \mathbf{v}^T(\lambda I - D)^{-1} \\ \mathbf{0} & I \end{bmatrix} \begin{bmatrix} \frac{1}{\rho} - \mathbf{v}^T(\lambda I - D)^{-1}\mathbf{v} & \mathbf{0}^T \\ \mathbf{0} & \lambda I - D \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^T \\ (\lambda I - D)^{-1}\mathbf{v} & I \end{bmatrix}.
\end{aligned}$$

These are simply block LDL^T factorizations of the first matrix. The first is the well-known one where the factorization is started with the $(1,1)$ block. The second is a 'backward' factorization that is started with the $(2,2)$ block. Because the determinants of the tridiagonal matrices are all unity, we have

$$\frac{1}{\rho}\det(\lambda I - D - \rho\mathbf{v}\mathbf{v}^T) = \frac{1}{\rho}(1 - \rho\mathbf{v}^T(\lambda I - D)^{-1}\mathbf{v})\det(\lambda I - D). \tag{4.20}$$

Denoting the eigenvalues of $D + \rho\mathbf{v}\mathbf{v}^T$ again by $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ this implies

$$\begin{aligned}
\prod_{j=1}^{n}(\lambda - \lambda_j) &= (1 - \rho\mathbf{v}^T(\lambda I - D)^{-1}\mathbf{v}) \prod_{j=1}^{n}(\lambda - d_j) \\
&= \left(1 - \rho\sum_{k=1}^{n}\frac{v_k^2}{\lambda - d_k}\right)\prod_{j=1}^{n}(\lambda - d_j) \\
&= \prod_{j=1}^{n}(\lambda - d_j) - \rho\sum_{k=1}^{n} v_k^2 \prod_{j\neq k}(\lambda - d_j)
\end{aligned} \tag{4.21}$$

Setting $\lambda = d_k$ gives

$$\prod_{j=1}^{n}(d_k - \lambda_j) = -\rho v_k^2 \prod_{\substack{j=1 \\ j\neq i}}^{n}(d_k - d_j) \tag{4.22}$$

or

$$
\begin{aligned}
v_k^2 &= \frac{-1}{\rho}\frac{\prod_{j=1}^{n}(d_k-\lambda_j)}{\prod_{\substack{j=1\\ j\neq i}}^{n}(d_k-d_j)} = \frac{-1}{\rho}\frac{\prod_{j=1}^{k-1}(d_k-\lambda_j)\prod_{j=k}^{n}(\lambda_j-d_k)(-1)^{n-k+1}}{\prod_{j=1}^{k-1}(d_k-d_j)\prod_{j=k+1}^{n}(d_j-d_k)(-1)^{n-k}} \\
&= \frac{1}{\rho}\frac{\prod_{j=1}^{k-1}(d_k-\lambda_j)}{\prod_{j=1}^{k-1}(d_k-d_j)}\frac{\prod_{j=k}^{n}(\lambda_j-d_k)}{\prod_{j=k+1}^{n}(d_j-d_k)} > 0.
\end{aligned} \tag{4.23}
$$

Therefore, the quantity on the right side is positive, so

$$
v_k = \sqrt{\frac{\prod_{j=1}^{k-1}(d_k-\lambda_j)\prod_{j=k}^{n}(\lambda_j-d_k)}{\rho\prod_{j=1}^{k-1}(d_k-d_j)\prod_{j=k+1}^{n}(d_j-d_k)}}. \tag{4.24}
$$

(Similar arguments hold if $\rho < 0$.) Thus, we have the solution of the following **inverse eigenvalue problem**:

> Given $D = \operatorname{diag}(d_1, \ldots, d_n)$ and values $\lambda_1, \ldots, \lambda_n$ that satisfy (4.18). Find a vector $\mathbf{v} = [v_1, \ldots, v_n]^T$ with positive components v_k such that the matrix $D + \mathbf{v}\mathbf{v}^T$ has the *prescribed* eigenvalues $\lambda_1, \ldots, \lambda_n$.

The solution is given by (4.24). The positivity of the v_k makes the solution unique.

4.6 Solving the secular equation

In this section we follow closely the exposition of Demmel [3]. We consider the computation of the zero of $f(\lambda)$ in the interval (d_i, d_{i+1}). We assume that $\rho = 1$.

We may simply apply Newton's iteration to solve $f(\lambda) = 0$. However, if we look carefully at Fig. 4.1 then we notice that the tangent at certain points in (d_i, d_{i+1}) crosses the real axis outside this interval. This happens in particular if the weights v_i or v_{i+1} are small. Therefore that zero finder has to be adapted in such a way that it captures the poles at the interval endpoints. It is relatively straightforward to try the ansatz

$$
h(\lambda) = \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} + c_3. \tag{4.25}
$$

Notice that, given the coefficients c_1, c_2, and c_3, the equation $h(\lambda) = 0$ can easily be solved by means of the equivalent quadratic equation

$$
c_1(d_{i+1} - \lambda) + c_2(d_i - \lambda) + c_3(d_i - \lambda)(d_{i+1} - \lambda) = 0. \tag{4.26}
$$

This equation has two zeros. Precisely one of them is inside (d_i, d_{i+1}).

The coefficients c_1, c_2, and c_3 are computed in the following way. Let us assume that we have available an approximation λ_j to the zero in (d_i, d_{i+1}). We request that $h(\lambda_j) = f(\lambda_j)$ and $h'(\lambda_j) = f'(\lambda_j)$. The exact procedure is as follows. We write

$$
f(\lambda) = 1 + \underbrace{\sum_{k=1}^{i}\frac{v_k^2}{d_k - \lambda}}_{\psi_1(\lambda)} + \underbrace{\sum_{k=i+1}^{n}\frac{v_k^2}{d_k - \lambda}}_{\psi_2(\lambda)} = 1 + \psi_1(\lambda) + \psi_2(\lambda). \tag{4.27}
$$

$\psi_1(\lambda)$ is a sum of positive terms and $\psi_2(\lambda)$ is a sum of negative terms. Both $\psi_1(\lambda)$ and $\psi_2(\lambda)$ can be computed accurately, whereas adding them would likely provoke cancellation and loss of relative accuracy. We now choose c_1 and $\hat{c}_1$ such that

$$h_1(\lambda) := \hat{c}_1 + \frac{c_1}{d_i - \lambda} \text{ satisfies } h_1(\lambda_j) = \psi_1(\lambda_j) \text{ and } h_1'(\lambda_j) = \psi_1'(\lambda_j). \tag{4.28}$$

This means that the graphs of h_1 and of ψ_1 are tangent at $\lambda = \lambda_j$. This is similar to Newton's method. However in Newton's method a straight line is fitted to the given function. The coefficients in (4.28) are given by

$$\begin{aligned} c_1 &= \psi_1'(\lambda_j)(d_i - \lambda_j)^2 > 0, \\ \hat{c}_1 &= \psi_1(\lambda_j) - \psi_1'(\lambda_j)(d_i - \lambda_j) = \sum_{k=1}^{i} v_k^2 \frac{d_k - d_i}{(d_k - \lambda_j)^2} \leq 0. \end{aligned}$$

Similarly, the two constants c_2 and $\hat{c}_2$ are determined such that

$$h_2(\lambda) := \hat{c}_2 + \frac{c_2}{d_{i+1} - \lambda} \text{ satisfies } h_2(\lambda_j) = \psi_2(\lambda_j) \text{ and } h_2'(\lambda_j) = \psi_2'(\lambda_j) \tag{4.29}$$

with the coefficients

$$\begin{aligned} c_2 &= \psi_2'(\lambda_j)(d_{i+1} - \lambda_j)^2 > 0, \\ \hat{c}_2 &= \psi_2(\lambda_j) - \psi_2'(\lambda_j)(d_{i+1} - \lambda_j) = \sum_{k=i+1}^{n} v_k^2 \frac{d_k - d_{i+1}}{(d_k - \lambda)^2} \geq 0. \end{aligned}$$

Finally, we set

$$h(\lambda) = 1 + h_1(\lambda) + h_2(\lambda) = \underbrace{(1 + \hat{c}_1 + \hat{c}_2)}_{c_3} + \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda}. \tag{4.30}$$

This zerofinder is converging quadratically to the desired zero [7]. Usually 2 to 3 steps are sufficient to get the zero to machine precision. Therefore finding a zero only requires $\mathcal{O}(n)$ flops. Thus, finding all zeros costs $\mathcal{O}(n^2)$ floating point operations.

4.7 A first algorithm

We are now ready to give the divide and conquer algorithm, see Algorithm 4.1.
All steps except step 10 require $\mathcal{O}(n^2)$ operations to complete. The step 10 costs n^3 flops. Thus, the full divide and conquer algorithm, requires

$$\begin{aligned} T(n) &= n^3 + 2 \cdot T(n/2) = n^3 + 2\left(\frac{n}{2}\right)^3 + 4T(n/4) \\ &= n^3 + \frac{n^3}{4} + 4\left(\frac{n}{4}\right)^3 + 8T(n/8) = \cdots = \frac{4}{3}n^3. \end{aligned} \tag{4.31}$$

This *serial* complexity of the algorithm very often overestimates the computational costs of the algorithm due to significant deflation that is observed surprisingly often.

Algorithm 4.1 The tridiagonal divide and conquer algorithm

Let $T \in \mathbb{C}^{n\times n}$ be a real symmetric tridiagonal matrix. This algorithm computes the spectral decomposition of $T = Q\Lambda Q^T$, where the diagonal Λ is the matrix of eigenvalues and Q is orthogonal.
if T is 1×1 **then**
  return $(\Lambda = T; Q = 1)$
else
  Partition $T = \begin{bmatrix} T_1 & O \\ O & T_2 \end{bmatrix} + \rho\mathbf{u}\mathbf{u}^T$ according to (4.2)
  Call this algorithm with T_1 as input and Q_1, Λ_1 as output.
  Call this algorithm with T_2 as input and Q_2, Λ_2 as output.
  Form $D + \rho\mathbf{v}\mathbf{v}^T$ from $\Lambda_1, \Lambda_2, Q_1, Q_2$ according to (4.4)–(4.6).
  Find the eigenvalues Λ and the eigenvectors Q' of $D + \rho\mathbf{v}\mathbf{v}^T$.
  Form $Q = \begin{bmatrix} Q_1 & O \\ O & Q_2 \end{bmatrix} \cdot Q'$ which are the eigenvectors of T.
  return $(\Lambda; Q)$
end if

4.7.1 A numerical example

Let A be a 4×4 matrix

$$A = D + \mathbf{v}\mathbf{v}^T = \begin{bmatrix} 0 & & & \\ & 2-\beta & & \\ & & 2+\beta & \\ & & & 5 \end{bmatrix} + \begin{bmatrix} 1 \\ \beta \\ \beta \\ 1 \end{bmatrix} \begin{bmatrix} 1 & \beta & \beta & 1 \end{bmatrix}. \tag{4.32}$$

In this example (that is similar to one in [8]) we want to point at a problem that the divide and conquer algorithm possesses as it is given in Algorithm 4.1, namely the loss of orthogonality among eigenvectors.

Before we do some MATLAB tests let us look more closely at D and $\mathbf{v}$ in (4.32). This example becomes difficult to solve if β gets very small. In Figures 4.2 to 4.5 we see graphs of the function $f_\beta(\lambda)$ that appears in the secular equation for $\beta = 1$, $\beta = 0.1$, and $\beta = 0.01$. The critical zeros move towards 2 from both sides. The weights $v_2^2 = v_3^2 = \beta^2$ are however not so small that they should be deflated.

The following MATLAB code shows the problem. We execute the commands for $\beta = 10^{-k}$ for $k = 0, 1, 2, 4, 8$.

```
v = [1 beta beta 1]';          % rank-1 modification
d = [0, 2-beta, 2+beta, 5]';   % diagonal matrix

L = eig(diag(d) + v*v')        % eigenvalues of the modified matrix
e = ones(4,1);
q = (d*e'-e*L').\(v*e');       % unnormalized eigenvectors cf. (5.15)

Q = sqrt(diag(q'*q));
q = q./(e*Q');                 % normalized eigenvectors

norm(q'*q-eye(4))              % check for orthogonality
```

We do not bother how we compute the eigenvalues. We simply use MATLAB's built-in function `eig`. We get the results of Table 4.1.

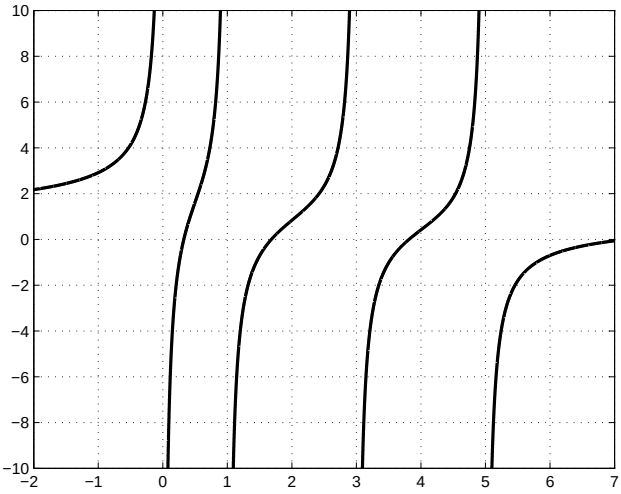

Figure 4.2: Secular equation corresponding to (4.32) for $\beta = 1$

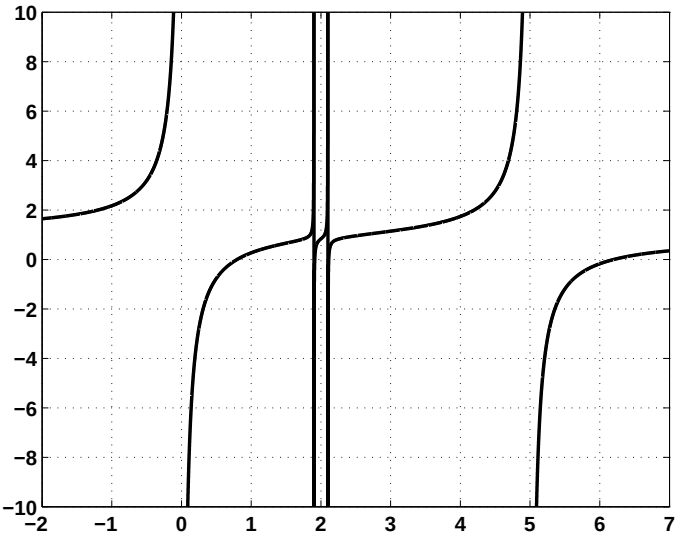

Figure 4.3: Secular equation corresponding to (4.32) for $\beta = 0.1$

We observe loss of orthogonality among the eigenvectors as the eigenvalues get closer and closer. This may not be surprising as we compute the eigenvectors by formula (4.15)

$$\mathbf{x} = \frac{(\lambda I - D)^{-1}\mathbf{v}}{\|(\lambda I - D)^{-1}\mathbf{v}\|}.$$

If $\lambda = \lambda_2$ and $\lambda = \lambda_3$ which are almost equal, $\lambda_2 \approx \lambda_3$ then intuitively one expects almost the same eigenvectors. We have in fact

$$Q^TQ - I_4 = \begin{bmatrix} -2.2204 \cdot 10^{-16} & 4.3553 \cdot 10^{-8} & 1.7955 \cdot 10^{-8} & -1.1102 \cdot 10^{-16} \\ 4.3553 \cdot 10^{-8} & 0 & -5.5511 \cdot 10^{-8} & -1.8298 \cdot 10^{-8} \\ 1.7955 \cdot 10^{-8} & -5.5511 \cdot 10^{-8} & -1.1102 \cdot 10^{-16} & -7.5437 \cdot 10^{-9} \\ -1.1102 \cdot 10^{-16} & -1.8298 \cdot 10^{-8} & -7.5437 \cdot 10^{-9} & 0 \end{bmatrix}.$$

Orthogonality is lost only with respect to the vectors corresponding to the eigenvalues close to 2.

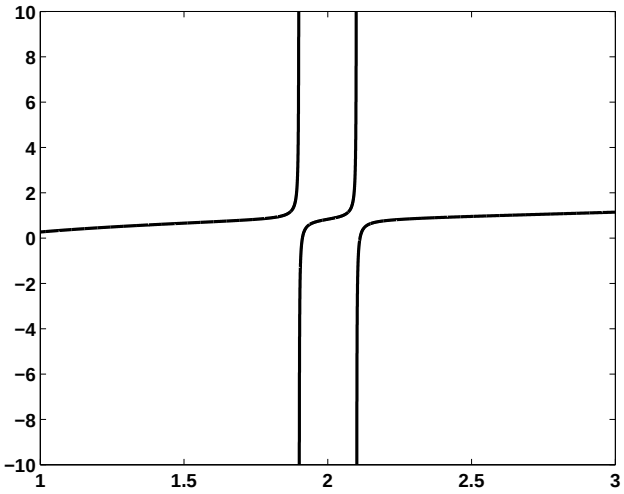

Figure 4.4: Secular equation corresponding to (4.32) for $\beta = 0.1$ for $1 \leq \lambda \leq 3$

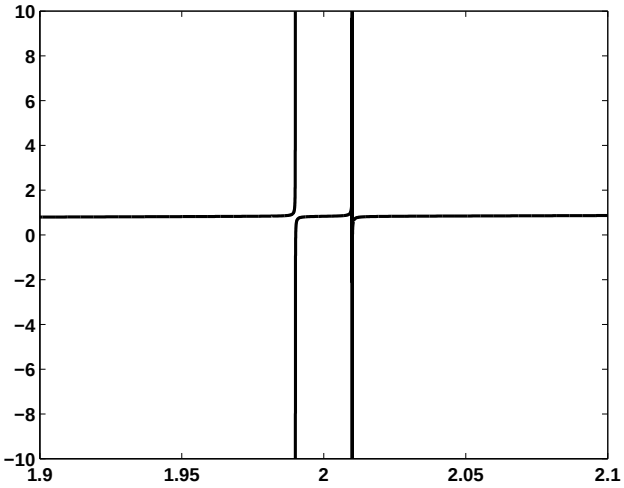

Figure 4.5: Secular equation corresponding to (4.32) for $\beta = 0.01$ for $1.9 \leq \lambda \leq 2.1$

Already Dongarra and Sorensen [4] analyzed this problem. In their formulation they normalize the vector $\mathbf{v}$ of $D + \rho\mathbf{v}\mathbf{v}^T$ to have norm unity, $\|\mathbf{v}\| = 1$. They stated

Lemma 4.1 *Let*

$$\mathbf{q}_\lambda^T = \left(\frac{v_1}{d_1 - \lambda}, \frac{v_2}{d_2 - \lambda}, \ldots, \frac{v_n}{d_n - \lambda}\right)\left[\frac{\rho}{f'(\lambda)}\right]^{1/2}. \tag{4.33}$$

Then for any $\lambda, \mu \notin \{d_1, \ldots, d_n\}$ we have

$$|\mathbf{q}_\lambda^T \mathbf{q}_\mu| = \frac{1}{|\lambda - \mu|}\frac{|f(\lambda) - f(\mu)|}{[f'(\lambda)f'(\mu)]^{1/2}}. \tag{4.34}$$

Proof. Observe that

$$\frac{\lambda - \mu}{(d_j - \lambda)(d_j - \mu)} = \frac{1}{d_j - \lambda} - \frac{1}{d_j - \mu}.$$

Then the proof is straightforward. ■

β	λ_1	λ_2	λ_3	λ_4	$\|Q^TQ - I\|$
1	0.325651	1.682219	3.815197	7.176933	$5.6674 \cdot 10^{-16}$
0.1	0.797024	1.911712	2.112111	6.199153	$3.4286 \cdot 10^{-15}$
0.01	0.807312	1.990120	2.010120	6.192648	$3.9085 \cdot 10^{-14}$
10^{-4}	0.807418	1.999900	2.000100	6.192582	$5.6767 \cdot 10^{-12}$
10^{-8}	0.807418	1.99999999000000	2.00000001000000	6.192582	$8.3188 \cdot 10^{-08}$

Table 4.1: Loss of orthogonality among the eigenvectors computed by (4.15)

Formula (4.34) indicates how problems may arise. In exact arithmetic, if λ and μ are eigenvalues then $f(\lambda) = f(\mu) = 0$. However, in floating point arithmetic this values may be small but nonzero, e.g., $\mathcal{O}(\varepsilon)$. If $|\lambda - \mu|$ is very small as well then we may have trouble! So, a remedy for the problem was for a long time to compute the eigenvalues in doubled precision, so that $f(\lambda) = \mathcal{O}(\varepsilon^2)$. This would counteract a potential $\mathcal{O}(\varepsilon)$ of $|\lambda - \mu|$.

This solution was quite unsatisfactory because doubled precision is in general very slow since it is implemented in software. It took a decade until a proper solution was found.

4.8 The algorithm of Gu and Eisenstat

Computing eigenvector according to the formula

$$\mathbf{x} = \alpha(\lambda I - D)^{-1}\mathbf{v} = \alpha \begin{pmatrix} \frac{v_1}{\lambda - d_1} \\ \vdots \\ \frac{v_n}{\lambda - d_n} \end{pmatrix}, \qquad \alpha = \|(\lambda I - D)^{-1}\mathbf{v}\|, \tag{4.35}$$

is bound to fail if λ is very close to a pole d_k and the difference $\lambda - d_k$ has an error of size $\mathcal{O}(\varepsilon|d_k|)$ instead of only $\mathcal{O}(\varepsilon|d_k - \lambda|)$. To resolve this problem Gu and Eisenstat [5] found a trick that is at the same time ingenious and simple.

They observed that the v_k in (4.24) are very accurately determined by the data d_i and λ_i. Therefore, once the eigenvalues are computed *accurately* a vector $\hat{\mathbf{v}}$ could be computed such that the λ_i are *accurate* eigenvalues of $D + \hat{\mathbf{v}}\hat{\mathbf{v}}$. If $\hat{\mathbf{v}}$ approximates well the original $\mathbf{v}$ then the new eigenvectors will be the exact eigenvectors of a slightly modified eigenvalue problem, which is all we can hope for.

The zeros of the secular equation can be computed accurately by the method presented in section 4.6. However, a shift of variables is necessary. In the interval (d_i, d_{i+1}) the origin of the real axis is moved to d_i if λ_i is closer to d_i than to d_{i+1}, i.e., if $f((d_i + d_{i+1})/2) > 0$. Otherwise, the origin is shifted to d_{i+1}. This shift of the origin avoids the computation of the smallest difference $d_i - \lambda$ (or $d_{i+1} - \lambda$) in (4.35), thus avoiding cancellation in this most sensitive quantity. Equation (4.26) can be rewritten as

$$\underbrace{(c_1\Delta_{i+1} + c_2\Delta_i + c_3\Delta_i\Delta_{i+1})}_{b} - \underbrace{(c_1 + c_2 + c_3(\Delta_i + \Delta_{i+1}))}_{-a}\eta + \underbrace{c_3}_{c}\eta^2 = 0, \tag{4.36}$$

where $\Delta_i = d_i - \lambda_j$, $\Delta_{i+1} = d_{i+1} - \lambda_j$, and $\lambda_{j+1} = \lambda_j + \eta$ is the next approximate zero. With equations (4.28)–(4.30) the coefficients in (4.36) get

$$\begin{aligned} a &= c_1 + c_2 + c_3(\Delta_i + \Delta_{i+1}) = (1 + \Psi_1 + \Psi_2)(\Delta_i + \Delta_{i+1}) - (\Psi_1' + \Psi_2')\Delta_i\Delta_{i+1}, \\ b &= c_1\Delta_{i+1} + c_2\Delta_i + c_3\Delta_i\Delta_{i+1} = \Delta_i\Delta_{i+1}(1 + \Psi_1 + \Psi_2), \\ c &= c_3 = 1 + \Psi_1 + \Psi_2 - \Delta_i\Psi_1' - \Delta_{i+1}\Psi_2'. \end{aligned} \tag{4.37}$$

If we are looking for a zero that is closer to d_i than to d_{i+1} then we move the origin to λ_j, i.e., we have e.g. $\Delta_i = -\lambda_j$. The solution of (4.36) that lies inside the interval is [7]

$$\eta = \begin{cases} \dfrac{a - \sqrt{a^2 - 4bc}}{2c}, & \text{if } a \le 0, \\ \dfrac{2b}{a + \sqrt{a^2 - 4bc}}, & \text{if } a > 0. \end{cases} \tag{4.38}$$

The following algorithm shows how step 9 of the tridiagonal divide and conquer algorithm 4.1 must be implemented.

Algorithm 4.2 A stable eigensolver for $D + \mathbf{v}\mathbf{v}^T$

1: This algorithm stably computes the spectral decomposition of $D+\mathbf{v}\mathbf{v}^T = Q\Lambda Q^T$ where $D = \mathrm{diag}(d_1, \ldots d_n)$, $\mathbf{v} = [v_1, \ldots, v_n] \in \mathbb{R}^n$, $\Lambda = \mathrm{diag}(\lambda_1, \ldots \lambda_n)$, and $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$.
2: $d_{i+1} = d_n + \|\mathbf{v}\|^2$.
3: In each interval (d_i, d_{i+1}) compute the zero λ_i of the secular equation $f(\lambda) = 0$.
4: Use the formula (4.24) to compute the vector $\hat{\mathbf{v}}$ such that the λ_i are the 'exact' eigenvalues of $D + \hat{\mathbf{v}}\hat{\mathbf{v}}$.
5: In each interval (d_i, d_{i+1}) compute the eigenvectors of $D + \hat{\mathbf{v}}\hat{\mathbf{v}}$ according to (4.15),

$$\mathbf{q}_i = \frac{(\lambda_i I - D)^{-1}\hat{\mathbf{v}}}{\|(\lambda_i I - D)^{-1}\hat{\mathbf{v}}\|}.$$

6: **return** $(\Lambda; Q)$

4.8.1 A numerical example [continued]

We continue the discussion of the example on page 86 where the eigenvalue problem of

$$A = D + \mathbf{v}\mathbf{v}^T = \begin{bmatrix} 0 & & & \\ & 2-\beta & & \\ & & 2+\beta & \\ & & & 5 \end{bmatrix} + \begin{bmatrix} 1 \\ \beta \\ \beta \\ 1 \end{bmatrix} \begin{bmatrix} 1 & \beta & \beta & 1 \end{bmatrix}. \tag{4.39}$$

The MATLAB code that we showed did not give orthogonal eigenvectors. We show in the following script that the formulae (4.24) really solve the problem.

```
dlam = zeros(n,1);
for k=1:n,
  [dlam(k), dvec(:,k)] = zerodandc(d,v,k);
end

V = ones(n,1);
for k=1:n,
  V(k) = prod(abs(dvec(k,:)))/prod(d(k) - d(1:k-1))/prod(d(k+1:n) - d(k));
  V(k) = sqrt(V(k));
end

Q = (dvec).\(V*e');
diagq = sqrt(diag(Q'*Q));
Q = Q./(e*diagq');
```

```
for k=1:n,
  if dlam(k)>0,
    dlam(k) = dlam(k) + d(k);
  else
    dlam(k) = d(k+1) + dlam(k);
  end
end

norm(Q'*Q-eye(n))
norm((diag(d) + v*v')*Q - Q*diag(dlam'))
```

A zero finder returns for each interval the quantity $\lambda_i - d_i$ and the vector $[d_1 - \lambda_i, \ldots, d_n - \lambda_i]^T$ to high precision. These vector elements have been computed as $(d_k - d_i) - (\lambda_i - d_i)$. The zerofinder of Li [7] has been employed here. At the end of this section we list the zerofinder written in MATLAB that was used here. The formulae (4.37) and (4.38) have been used to solve the quadratic equation (4.36). Notice that only one of the `while` loops is traversed, depending on if the zero is closer to the pole on the left or to the right of the interval. The v_k of formula (4.24) are computed next. Q contains the eigenvectors.

β	Algorithm	$\|Q^TQ - I\|$	$\|AQ - Q\Lambda\|$
0.1	I	$3.4286 \cdot 10^{-15}$	$5.9460 \cdot 10^{-15}$
	II	$2.2870 \cdot 10^{-16}$	$9.4180 \cdot 10^{-16}$
0.01	I	$3.9085 \cdot 10^{-14}$	$6.9376 \cdot 10^{-14}$
	II	$5.5529 \cdot 10^{-16}$	$5.1630 \cdot 10^{-16}$
10^{-4}	I	$5.6767 \cdot 10^{-12}$	$6.3818 \cdot 10^{-12}$
	II	$2.2434 \cdot 10^{-16}$	$4.4409 \cdot 10^{-16}$
10^{-8}	I	$8.3188 \cdot 10^{-08}$	$1.0021 \cdot 10^{-07}$
	II	$2.4980 \cdot 10^{-16}$	$9.4133 \cdot 10^{-16}$

Table 4.2: Loss of orthogonality among the eigenvectors computed by the straightforward algorithm (I) and the Gu-Eisenstat approach (II)

Again we ran the code for $\beta = 10^{-k}$ for $k = 0, 1, 2, 4, 8$. The numbers in Table 4.2 confirm that the new formulae are much more accurate than the straight forward ones. The norms of the errors obtained for the Gu-Eisenstat algorithm always are in the order of machine precision, i.e., 10^{-16}.

```
function [lambda,dl] = zerodandc(d,v,i)
% ZERODANDC - Computes eigenvalue lambda in the i-th interval
%             (d(i), d(i+1)) with Li's 'middle way' zero finder
%             dl is the n-vector [d(1..n) - lambda]

n = length(d);
di = d(i);
v = v.^2;
if i < n,
  di1 = d(i+1);  lambda = (di + di1)/2;
else
  di1 = d(n) + norm(v)^2; lambda = di1;
end
eta = 1;
psi1 = sum((v(1:i).^2)./(d(1:i) - lambda));
```

```
psi2 = sum((v(i+1:n).^2)./(d(i+1:n) - lambda));

if  1 + psi1 + psi2 > 0,  %  zero is on the left half of the interval

  d = d - di;  lambda = lambda - di; di1 = di1 - di; di = 0;

  while  abs(eta) > 10*eps
    psi1 = sum(v(1:i)./(d(1:i) - lambda));
    psi1s = sum(v(1:i)./((d(1:i) - lambda)).^2);

    psi2 = sum((v(i+1:n))./(d(i+1:n) - lambda));
    psi2s = sum(v(i+1:n)./((d(i+1:n) - lambda)).^2);

    % Solve for zero
    Di = -lambda; Di1 = di1 - lambda;
    a = (Di + Di1)*(1 + psi1 + psi2) - Di*Di1*(psi1s + psi2s);
    b = Di*Di1*(1 + psi1 + psi2);
    c = (1 + psi1 + psi2) - Di*psi1s - Di1*psi2s;
    if a > 0,
      eta = (2*b)/(a + sqrt(a^2 - 4*b*c));
    else
      eta = (a - sqrt(a^2 - 4*b*c))/(2*c);
    end
    lambda = lambda + eta;
  end

else  %  zero is on the right half of the interval

  d = d - di1; lambda = lambda - di1; di = di - di1; di1 = 0;

  while  abs(eta) > 10*eps
    psi1 = sum(v(1:i)./(d(1:i) - lambda));
    psi1s = sum(v(1:i)./((d(1:i) - lambda)).^2);

    psi2 = sum((v(i+1:n))./(d(i+1:n) - lambda));
    psi2s = sum(v(i+1:n)./((d(i+1:n) - lambda)).^2);

    % Solve for zero
    Di = di - lambda; Di1 = - lambda;
    a = (Di + Di1)*(1 + psi1 + psi2) - Di*Di1*(psi1s + psi2s);
    b = Di*Di1*(1 + psi1 + psi2);
    c = (1 + psi1 + psi2) - Di*psi1s - Di1*psi2s;
    if a > 0,
      eta = (2*b)/(a + sqrt(a^2 - 4*b*c));
    else
      eta = (a - sqrt(a^2 - 4*b*c))/(2*c);
    end
    lambda = lambda + eta;
  end

end

dl = d - lambda;

return
```

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide - Release 2.0*, SIAM, Philadelphia, PA, 1994. (Software and guide are available from Netlib at URL http://www.netlib.org/lapack/).

[2] J. J. M. Cuppen, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[3] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

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

[5] M. Gu and S. C. Eisenstat, *A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1266–1276.

[6] ——, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

[7] R.-C. Li, *Solving secular equations stably and efficiently*, Technical Report UT-CS-94-260, University of Tennessee, Knoxville, TN, Nov. 1994. LAPACK Working Note No. 89.

[8] D. C. Sorensen and P. T. P. Tang, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM J. Numer. Anal., 28 (1991), pp. 1752–1775.