Chapter 3

Newton methods

3.1 Linear and nonlinear eigenvalue problems

In **linear** eigenvalue problems we have to find values $\lambda \in \mathbb{C}$ such that $\lambda I - A$ is **singular**. Here $A \in \mathbb{F}^{n\times n}$ is a given real or complex matrix. Equivalently, we have to find values $\lambda \in \mathbb{C}$ such that there is a nontrivial (nonzero) $\mathbf{x}$ that satisfies

$$(A - \lambda I)\mathbf{x} = \mathbf{0} \quad \Longleftrightarrow \quad A\mathbf{x} = \lambda\mathbf{x}. \tag{3.1}$$

In the linear eigenvalue problem (3.1) the eigenvalue λ appears linearly. However, as the unknown λ is multiplied with the unknown vector $\mathbf{x}$, the problem is in fact nonlinear. We have $n+1$ unknowns $\lambda, x_1, \ldots, x_n$ that are not uniquely defined by the n equations in (3.1). We have noticed earlier, that the *length* of the eigenvector $\mathbf{x}$ is not determined. This can be rectified by adding a further equation that fixes the length of $\mathbf{x}$. The straightforward condition is

$$\|\mathbf{x}\|^2 = \mathbf{x}^*\mathbf{x} = 1, \tag{3.2}$$

that determines $\mathbf{x}$ up to a complex scalar of modulus 1, in the real case ± 1. Another condition to normalize $\mathbf{x}$ is by requesting that

$$\mathbf{c}^T\mathbf{x} = 1, \qquad \text{for some } \mathbf{c}. \tag{3.3}$$

Eq. (3.3) is linear in $\mathbf{x}$ and thus simpler. However, the combined equations (3.1)–(3.3) are nonlinear anyway. Furthermore, $\mathbf{c}$ must be chosen such that it has a strong component in the (unknown) direction of the searched eigenvector. This requires some knowledge about the solution.

In **nonlinear** eigenvalue problems we have to find values $\lambda \in \mathbb{C}$ such that

$$A(\lambda)\mathbf{x} = \mathbf{0} \tag{3.4}$$

where $A(\lambda)$ is a matrix the elements of which depend on λ in a nonlinear way. An example is a matrix polynomial,

$$A(\lambda) = \sum_{k=0}^{d} \lambda^k A_k, \qquad A_k \in \mathbb{F}^{n\times n}. \tag{3.5}$$

The linear eigenvalue problem (3.1) is a special case with $d = 1$,

$$A(\lambda) = A_0 - \lambda A_1, \qquad A_0 = A, \quad A_1 = I.$$

Quadratic eigenvalue problems of the form

$$A\mathbf{x} + \lambda K\mathbf{x} + \lambda^2 M\mathbf{x} = \mathbf{0}. \tag{3.6}$$

Matrix polynomials can be linearized, i.e., they can be transformed in a linear eigenvalue of bigger size. The quadratic eigenvalue problem (3.6) can be transformed in a linear eigenvalue problem of size $2n$. Setting $\mathbf{y} = \lambda\mathbf{x}$ we get

$$\begin{pmatrix} A & O \\ O & I \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} -K & -M \\ I & O \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$$

or

$$\begin{pmatrix} A & K \\ O & I \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \lambda \begin{pmatrix} O & -M \\ I & O \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}.$$

Notice that many other linearizations are possible [2, 6]. Notice also the relation with the transformation of high order to first order ODE's [5, p. 478].

Instead of looking at the nonlinear system (3.1) (complemented with (3.3) or (3.2)) we may look at the nonlinear scalar equation

$$f(\lambda) := \det\ A(\lambda) = 0 \tag{3.7}$$

and apply some zero finder. Here the question arises how to compute $f(\lambda)$ and in particular $f'(\lambda) = \frac{d}{d\lambda}\det\ A(\lambda)$.

3.2 Zeros of the determinant

We first consider the computation of eigenvalues and subsequently eigenvectors by means of computing zeros of the determinant $\det(A(\lambda))$.

Gaussian elimination with partial pivoting (GEPP) applied to $A(\lambda)$ provides the decomposition

$$P(\lambda)A(\lambda) = L(\lambda)U(\lambda), \tag{3.8}$$

where $P(\lambda)$ is the permutation matrix due to partial pivoting, $L(\lambda)$ is a lower unit triangular matrix, and $U(\lambda)$ is an upper triangular matrix. From well-known properties of the determinant function, equation (3.8) gives

$$\det P(\lambda) \cdot \det A(\lambda) = \det L(\lambda) \cdot \det U(\lambda).$$

Taking the particular structures of the factors in (3.8) into account, we get

$$f(\lambda) = \det A(\lambda) = \pm 1 \cdot \prod_{i=1}^{n} u_{ii}(\lambda). \tag{3.9}$$

The derivative of $\det A(\lambda)$ is

$$\begin{aligned} f'(\lambda) &= \pm 1 \cdot \sum_{i=1}^{n} u'_{ii}(\lambda) \prod_{j\neq i}^{n} u_{jj}(\lambda) \\ &= \pm 1 \cdot \sum_{i=1}^{n} \frac{u'_{ii}(\lambda)}{u_{ii}(\lambda)} \prod_{j=1}^{n} u_{jj}(\lambda) = \sum_{i=1}^{n} \frac{u'_{ii}(\lambda)}{u_{ii}(\lambda)} f(\lambda). \end{aligned} \tag{3.10}$$

How can we compute the derivatives u'_{ii} of the diagonal elements of $U(\lambda)$?

3.2.1 Algorithmic differentiation

A clever way to compute derivatives of a function is by **algorithmic differentiation**, see e.g., [1]. Here we assume that we have an algorithm available that computes the value $f(\lambda)$ of a function f, given the input argument λ. By algorithmic differentiation a new algorithm is obtained that computes besides $f(\lambda)$ the derivative $f'(\lambda)$.

The idea is easily explained by means of the Horner scheme to evaluate polynomials. Let

$$f(z) = \sum_{i=1}^{n} c_i z^i.$$

be a polynomial of degree n. $f(z)$ can be written in the form

$$f(z) = c_0 + z\,(c_1 + z\,(c_2 + \cdots + z\,(c_n)\cdots))$$

which gives rise to the recurrence

$$\begin{aligned} p_n &:= c_n, \\ p_i &:= z\,p_{i+1} + c_i, \qquad i = n-1, n-2, \ldots, 0, \\ f(z) &:= p_0. \end{aligned}$$

Note that each of the p_i can be considered as a function (polynomial) in z. We use the above recurrence to determine the derivatives dp_i,

$$\begin{aligned} dp_n &:= 0, \quad p_n := c_n, \\ dp_i &:= p_{i+1} + z\,dp_{i+1}, \quad p_i := z\,p_{i+1} + c_i, \quad i = n-1, n-2, \ldots, 0, \\ f'(z) &:= dp_0, \quad f(z) := p_0. \end{aligned}$$

We can proceed in a similar fashion for computing $\det A(\lambda)$. We however need to be able to compute the derivatives a'_{ij}. Then, we can derive each single assignment in the GEPP algorithm.

If we restrict ourselves to the standard eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$ then $A(\lambda) = A - \lambda I$. Then, $a'_{ij} = \delta_{ij}$, the Kronecker δ.

3.2.2 Hyman's algorithm

In a Newton iteration we have to compute the determinant for possibly many values λ. Using the factorization (3.8) leads to computational costs of $\frac{2}{3}n^3$ flops (floating point operations) for each factorization, i.e., per iteration step. If this algorithm was used to compute all eigenvalues then an excessive amount of flops would be required. Can we do better?

The strategy is to transform A by a similarity transformation to a **Hessenberg matrix**, i.e., a matrix H whose entries below the lower off-diagonal are zero,

$$h_{ij} = 0, \qquad i > j + 1.$$

Any matrix A can be transformed into a similar Hessenberg matrix H by means of a sequence of elementary unitary matrices called **Householder transformations**. The details are given in Section 4.3.

Let $S^*AS = H$, where S is unitary. S is the product of the just mentioned Householder transformations. Then

$$A\mathbf{x} = \lambda\mathbf{x} \quad \Longleftrightarrow \quad H\mathbf{y} = \lambda\mathbf{y}, \qquad \mathbf{x} = S\mathbf{y}.$$

So, A and H have equal eigenvalues (A and H are similar) and the eigenvectors are transformed by S. We now assume that H is **unreduced**, i.e., $h_{i+1,i} \neq 0$ for all i. Otherwise we can split $H\mathbf{x} = \lambda\mathbf{x}$ in smaller problems.

Let λ be an eigenvalue of H and

$$(H - \lambda I)\mathbf{x} = \mathbf{0}, \tag{3.11}$$

i.e., $\mathbf{x}$ is an eigenvector of H associated with the eigenvalue λ. Then the last component of $\mathbf{x}$ cannot be zero, $x_n \neq 0$. The proof is by contradiction. Let $x_n = 0$. Then (for $n = 4$)

$$\begin{pmatrix} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The last equation reads

$$h_{n,n-1}x_{n-1} + (h_{nn} - \lambda) \cdot 0 = 0$$

from which $x_{n-1} = 0$ follows since we assumed $h_{n,n-1} \neq 0$. In the exact same procedure we obtain $x_{n-2} = 0, \ldots, x_1 = 0$. But the zero vector cannot be an eigenvector. Therefore, x_n must not be zero. Without loss of generality we can set $x_n = 1$.

We continue to expose the procedure with a problem size $n = 4$. If λ is an eigenvalue then there are x_i, $1 \le i < n$, such that

$$\begin{pmatrix} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.12}$$

If λ is **not** an eigenvalue then we determine the x_i such that

$$\left(\begin{array}{ccc|c} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ \hline h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{array}\right) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p(\lambda) \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.13}$$

We determine the $n-1$ numbers $x_{n-1}, x_{n-2}, \ldots, x_1$ by

$$x_i = \frac{-1}{h_{i+1,i}}\big((h_{i+1,i+1} - \lambda)\, x_{i+1} + h_{i+1,i+2}\, x_{i+2} + \cdots + h_{i+1,n} \underbrace{x_n}_{1}\big), \quad i = n-1, \ldots, 1.$$

The x_i are functions of λ, in fact, $x_i \in \mathbb{P}_{n-i}$. The first equation in (3.13) gives

$$(h_{1,1} - \lambda)\, x_1 + h_{1,2}\, x_2 + \cdots + h_{1,n}\, x_n = p(\lambda). \tag{3.14}$$

Eq. (3.13) can be understood by the factorization

$$\begin{pmatrix} h_{11}-\lambda & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22}-\lambda & h_{23} & h_{24} \\ & h_{32} & h_{33}-\lambda & h_{34} \\ & & h_{43} & h_{44}-\lambda \end{pmatrix} \begin{pmatrix} 1 & & & x_1 \\ & 1 & & x_2 \\ & & 1 & x_3 \\ & & & 1 \end{pmatrix}$$

$$= \begin{pmatrix} h_{11}-\lambda & h_{12} & h_{13} & p(\lambda) \\ h_{21} & h_{22}-\lambda & h_{23} & 0 \\ & h_{32} & h_{33}-\lambda & 0 \\ & & h_{43} & 0 \end{pmatrix}.$$

The last column of this equation corresponds to (3.13). Taking determinants yields

$$\det(H-\lambda I) = (-1)^{n-1}\left(\prod_{i=1}^{n-1} h_{i+1,i}\right) p(\lambda) = c\cdot p(\lambda).$$

So, $p(\lambda)$ is a constant multiple of the determinant of $H-\lambda I$. Therefore, we can solve $p(\lambda)=0$ instead of $\det(H-\lambda I)=0$.

Since the quantities $x_1, x_2, \ldots, x_n$ and thus $p(\lambda)$ are differentiable functions of λ, we can algorithmically differentiate to get $p'(\lambda)$.

For $i = n-1, \ldots, 1$ we have

$$x_i' = \frac{-1}{h_{i+1,i}}\big(-x_{i+1} + (h_{i+1,i+1}-\lambda)\,x_{i+1}' + h_{i+1,i+2}\,x_{i+2} + \cdots + h_{i+1,n-1}x_{n-1}'\big).$$

Finally,

$$c\cdot f'(\lambda) = -x_1 + (h_{1,n}-\lambda)\,x_1' + h_{1,2}\,x_2 + \cdots + h_{1,n-1}x_{n-1}'.$$

Algorithm 3.2.2 implements Hyman's algorithm that returns $p(\lambda)$ and $p'(\lambda)$ given an input parameter λ [7].

Algorithm 3.1 Hyman's algorithm

```
1: Choose a value $\lambda$.
2: $x_n := 1$; $dx_n := 0$;
3: for $i = n-1$ downto 1 do
4:    $s = (\lambda - h_{i+1,i+1})\,x_{i+1}$; $ds = x_{i+1} + (\lambda - h_{i+1,i+1})\,dx_{i+1}$;
5:    for $j = i+1$ to $n$ do
6:       $s = s - h_{i+1,j}x_j$; $ds = ds - h_{i+1,j}dx_j$;
7:    end for
8:    $x_i = s/h_{i+1,i}$; $dx_i = ds/h_{i+1,i}$;
9: end for
10: $s = -(\lambda - h_{1,1})x_1$; $ds = -(\lambda - h_{1,1})dx_1 - x_1$;
11: for $i = 2$ to $n$ do
12:    $s = s + h_{1,i}x_i$; $ds = ds + h_{1,i}dx_i$;
13: end for
14: $p(\lambda) := s$; $p'(\lambda) := ds$;
```

This algorithm computes $p(\lambda) = c'\cdot\det(H(\lambda))$ and its derivative $p'(\lambda)$ of a Hessenberg matrix H in $\mathcal{O}(n^2)$ operations. Inside a Newton iteration the new iterate would be obtained by

$$\lambda_{k+1} = \lambda_k - \frac{p(\lambda_k)}{p'(\lambda_k)}, \qquad k = 0, 1, \ldots$$

The factor c' cancels. An initial guess λ_0 has to be chosen to start the iteration. It is clear that a good guess reduces the iteration count of the Newton method. The iteration is considered converged if $f(\lambda_k) \approx 0$. The vector $\mathbf{x} = (x_1, x_2, \ldots, x_{n-1}, 1)^T$ is a good approximation of the corresponding eigenvector.

Remark 3.1. Higher order deriatives of f can be computed in an analogous fashion. Higher order zero finders (e.g. Laguerre's zero finder) are then applicable [3]. □

3.2.3 Computing multiple zeros

If we have found a zero z of $f(x) = 0$ and want to compute another one, we want to avoid recomputing the already found z.

We can **explicitly deflate** the zero by defining a new function

$$f_1(x) := \frac{f(x)}{x - z}, \tag{3.15}$$

and apply our method of choice to f_1. This procedure can in particular be done with polynomials. The coefficients of f_1 are however very sensitive to inaccuracies in z. We can proceed similarly for multiple zeros $z_1, \ldots, z_m$. Explicit deflation is not recommended and often not feasible since f is not given explicitely.

For the reciprocal Newton correction for f_1 in (3.15) we get

$$\frac{f_1'(x)}{f_1(x)} = \frac{\frac{f'(x)}{x-z} - \frac{f(x)}{(x-z)^2}}{\frac{f(x)}{x-z}} = \frac{f'(x)}{f(x)} - \frac{1}{x-z}.$$

Then a Newton correction becomes

$$x^{(k+1)} = x_k - \cfrac{1}{\cfrac{f'(x_k)}{f(x_k)} - \cfrac{1}{x_k - z}} \tag{3.16}$$

and similarly for multiple zeros $z_1, \ldots, z_m$. Working with (3.16) is called **implicit deflation**. Here, f is not modified. In this way errors in z are not propagated to f_1

3.3 Newton methods for the constrained matrix problem

We consider the nonlinear eigenvalue problem (3.4) equipped with the normalization condition (3.3),

$$\begin{aligned} T(\lambda)\,\mathbf{x} &= \mathbf{0}, \\ \mathbf{c}^T\mathbf{x} &= 1, \end{aligned} \tag{3.17}$$

where $\mathbf{c}$ is some given vector. At a solution $(\mathbf{x}, \lambda)$, $\mathbf{x} \neq \mathbf{0}$, $T(\lambda)$ is singular. Note that $\mathbf{x}$ is defined only up to a (nonzero) multiplicative factor. $\mathbf{c}^T\mathbf{x} = 1$ is just a way to normalize $\mathbf{x}$. Another one would be $\|\mathbf{x}\|_2 = 1$, cf. the next section.

Solving (3.17) is equivalent with finding a zero of the nonlinear function $\mathbf{f}(\mathbf{x}, \lambda)$,

$$\mathbf{f}(\mathbf{x}, \lambda) = \begin{pmatrix} T(\lambda)\,\mathbf{x} \\ \mathbf{c}^T\mathbf{x} - 1 \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}. \tag{3.18}$$

To apply Newton's zero finding method we need the Jacobian of $\mathbf{f}$,

$$J(\mathbf{x}, \lambda) \equiv \frac{\partial \mathbf{f}(\mathbf{x}, \lambda)}{\partial(\mathbf{x}, \lambda)} = \begin{pmatrix} T(\lambda) & T'(\lambda)\mathbf{x} \\ \mathbf{c}^T & 0 \end{pmatrix}. \tag{3.19}$$

Here, $T'(\lambda)$ denotes the (elementwise) derivative of T with respect to λ. Then, a step of Newton's iteration is given by

$$\begin{pmatrix}\mathbf{x}_{k+1}\\ \lambda_{k+1}\end{pmatrix} = \begin{pmatrix}\mathbf{x}_{k}\\ \lambda_{k}\end{pmatrix} - J(\mathbf{x}_k,\lambda_k)^{-1}\mathbf{f}(\mathbf{x}_k,\lambda_k), \tag{3.20}$$

or, with the abbreviations $T_k := T(\lambda_k)$ and $T'_k := T'(\lambda_k)$,

$$\begin{pmatrix}T_k & T'_k\,\mathbf{x}_k\\ \mathbf{c}^T & 0\end{pmatrix}\begin{pmatrix}\mathbf{x}_{k+1}-\mathbf{x}_k\\ \lambda_{k+1}-\lambda_k\end{pmatrix} = \begin{pmatrix}-T_k\,\mathbf{x}_k\\ 1-\mathbf{c}^T\mathbf{x}_k\end{pmatrix}. \tag{3.21}$$

If $\mathbf{x}_k$ is normalized, $\mathbf{c}^T\mathbf{x}_k = 1$, then the second equation in (3.21) yields

$$\mathbf{c}^T(\mathbf{x}_{k+1}-\mathbf{x}_k) = 0 \quad\Longleftrightarrow\quad \mathbf{c}^T\mathbf{x}_{k+1} = 1. \tag{3.22}$$

The first equation in (3.21) gives

$$T_k\,(\mathbf{x}_{k+1}-\mathbf{x}_k) + (\lambda_{k+1}-\lambda_k)\,T'_k\,\mathbf{x}_k = -T_k\,\mathbf{x}_k \quad\Longleftrightarrow\quad T_k\,\mathbf{x}_{k+1} = -(\lambda_{k+1}-\lambda_k)\,T'_k\,\mathbf{x}_k.$$

We introduce the auxiliary vector $\mathbf{u}_{k+1}$ by

$$T_k\,\mathbf{u}_{k+1} = T'_k\,\mathbf{x}_k. \tag{3.23}$$

Note that

$$\mathbf{x}_{k+1} = -(\lambda_{k+1}-\lambda_k)\,\mathbf{u}_{k+1}. \tag{3.24}$$

So, $\mathbf{u}_{k+1}$ points in the desired direction; it just needs to be normalized. Premultiplying (3.24) by $\mathbf{c}^T$ and using (3.22) gives

$$1 = \mathbf{c}^T\mathbf{x}_{k+1} = -(\lambda_{k+1}-\lambda_k)\,\mathbf{c}^T\mathbf{u}_{k+1},$$

or

$$\lambda_{k+1} = \lambda_k - \frac{1}{\mathbf{c}^T\mathbf{u}_{k+1}}. \tag{3.25}$$

In summary, we get the following procedure.

Algorithm 3.2 Newton iteration for solving (3.18)

```
1: Choose a starting vector $\mathbf{x}_0 \in \mathbb{R}^n$ with $\mathbf{c}^T\mathbf{x}_0 = 1$. Set $k := 0$.
2: repeat
3:   Solve $T(\lambda_k)\,\mathbf{u}_{k+1} := T'(\lambda_k)\,\mathbf{x}_k$ for $\mathbf{u}_{k+1}$;        (3.23)
4:   $\mu_k := \mathbf{c}^T\mathbf{u}_{k+1}$;
5:   $\mathbf{x}_{k+1} := \mathbf{u}_{k+1}/\mu_k$;        (Normalize $\mathbf{u}_{k+1}$)
6:   $\lambda_{k+1} := \lambda_k - 1/\mu_k$;        (3.25)
7:   $k := k+1$;
8: until some convergence criterion is satisfied
```

If the linear eigenvalue problem is solved by Algorithm 3.3 then $T'(\lambda)\mathbf{x} = \mathbf{x}$. In each iteration step a linear system has to be solved which requires the factorization of a matrix.

We now change the way we normalize $\mathbf{x}$. Problem (3.17) becomes

$$T(\lambda)\,\mathbf{x} = \mathbf{0}, \qquad \|\mathbf{x}\|^2 = 1, \tag{3.26}$$

with the corresponding nonlinear system of equations

$$\mathbf{f}(\mathbf{x},\lambda) = \begin{pmatrix} T(\lambda)\,\mathbf{x} \\ \frac{1}{2}(\mathbf{x}^T\mathbf{x}-1) \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 0 \end{pmatrix}. \tag{3.27}$$

The Jacobian now is

$$J(\mathbf{x},\lambda) \equiv \frac{\partial \mathbf{f}(\mathbf{x},\lambda)}{\partial(\mathbf{x},\lambda)} = \begin{pmatrix} T(\lambda) & T'(\lambda)\,\mathbf{x} \\ \mathbf{x}^T & 0 \end{pmatrix}. \tag{3.28}$$

The Newton step (3.20) is changed into

$$\begin{pmatrix} T_k & T_k'\,\mathbf{x}_k \\ \mathbf{x}_k^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x}_{k+1}-\mathbf{x}_k \\ \lambda_{k+1}-\lambda_k \end{pmatrix} = \begin{pmatrix} -T_k\,\mathbf{x}_k \\ \frac{1}{2}(1-\mathbf{x}_k^T\mathbf{x}_k) \end{pmatrix}. \tag{3.29}$$

If $\mathbf{x}_k$ is normalized, $\|\mathbf{x}_k\| = 1$, then the second equation in (3.29) gives

$$\mathbf{x}_k^T(\mathbf{x}_{k+1}-\mathbf{x}_k) = 0 \quad \Longleftrightarrow \quad \mathbf{x}_k^T\mathbf{x}_{k+1} = 1. \tag{3.30}$$

The correction $\Delta\mathbf{x}_k := \mathbf{x}_{k+1}-\mathbf{x}_k$ is orthogonal to the actual approximation. The first equation in (3.29) is the same as in (3.21). Again, we employ the auxiliary vector $\mathbf{u}_{k+1}$ defined in (3.23). Premultiplying (3.24) by $\mathbf{x}_k^T$ and using (3.30) gives

$$1 = \mathbf{x}_k^T\mathbf{x}_{k+1} = -(\lambda_{k+1}-\lambda_k)\,\mathbf{x}_k^T\mathbf{u}_{k+1},$$

or

$$\lambda_{k+1} = \lambda_k - \frac{1}{\mathbf{x}_k^T\mathbf{u}_{k+1}}. \tag{3.31}$$

The next iterate $\mathbf{x}_{k+1}$ is obtained by normalizing $\mathbf{u}_{k+1}$,

$$\mathbf{x}_{k+1} = \mathbf{u}_{k+1}/\|\mathbf{u}_{k+1}\|. \tag{3.32}$$

Algorithm 3.3 Newton iteration for solving (3.27)

```
1: Choose a starting vector $\mathbf{x}_0 \in \mathbb{R}^n$ with $\|\mathbf{x}^{(0)}\| = 1$. Set $k := 0$.
2: repeat
3:    Solve $T(\lambda_k)\,\mathbf{u}_{k+1} := T'(\lambda_k)\,\mathbf{x}_k$ for $\mathbf{u}_{k+1}$;          (3.23)
4:    $\mu_k := \mathbf{x}_k^T\mathbf{u}_{k+1}$;
5:    $\lambda_{k+1} := \lambda_k - 1/\mu_k$;                                          (3.31)
6:    $\mathbf{x}_{k+1} := \mathbf{u}_{k+1}/\|\mathbf{u}_{k+1}\|$;                     (Normalize $\mathbf{u}_{k+1}$)
7:    $k := k+1$;
8: until some convergence criterion is satisfied
```

3.4 Successive linear approximations

Ruhe [4] suggested the following method which is not derived as a Newton method. It is based on an expansion of $T(\lambda)$ at some approximate eigenvalue λ_k.

$$T(\lambda)\mathbf{x} \approx (T(\lambda_k) - \vartheta T'(\lambda_k))\mathbf{x} = \mathbf{0}, \qquad \lambda = \lambda_k - \vartheta. \tag{3.33}$$

Algorithm 3.4 Algorithm of successive linear problems

```
1: Start with approximation $\lambda_1$ of an eigenvalue of $T(\lambda)$.
2: for $k = 1, 2, \ldots$ do
3:    Solve the linear eigenvalue problem $T(\lambda)\mathbf{u} = \vartheta T'(\lambda)\mathbf{u}$.
4:    Choose an eigenvalue $\vartheta$ smallest in modulus.
5:    $\lambda_{k+1} := \lambda_k - \vartheta$;
6: end for
```

Equation (3.33) is a generalized eigenvalue problem with eigenvalue ϑ. If λ_k is a good approximation of an eigenvalue, then it is straightforward to compute the smallest eigenvalue ϑ of

$$T(\lambda_k)\mathbf{x} = \vartheta\, T'(\lambda_k)\mathbf{x} \tag{3.34}$$

and update λ_k by $\lambda_{k+1} = \lambda_k - \vartheta$.

Remark: If T is twice continuously differentiable, and λ is an eigenvalue of problem (1) such that $T'(\lambda)$ is singular and 0 is an algebraically simple eigenvalue of $T'(\lambda)^{-1}T(\lambda)$, then the method in Algorithm 3.4 converges quadratically towards λ.

Bibliography

[1] P. Arbenz and W. Gander, *Solving nonlinear eigenvalue problems by algorithmic differentiation*, Computing, 36 (1986), pp. 205–215.

[2] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Structured polynomial eigenvalue problems: Good vibrations from good linearizations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 1029–1051.

[3] B. Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Math. Comput., 18 (1964), pp. 464–485.

[4] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 10 (1973), pp. 674–689.

[5] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, 1986.

[6] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.

[7] R. C. Ward, *The QR algorithm and Hyman's method on vector computers*, Math. Comput., 30 (1976), pp. 132–142.