Theorem 6.3 *Let the eigenvalues λ_i of A be ordered decreasingly. Then the angle $\theta(u_i, \mathcal{K}_m)$ between the exact eigenvector u_i associated with λ_i and the $m-th$ Krylov subspace $\mathcal{K}_m$ satisfies the inequality,*

$$\tan\theta(u_i, \mathcal{K}_m) \leq \frac{\kappa_i}{C_{m-i}(1+2\gamma_i)} \tan\theta(v_1, u_i) \ , \tag{6.39}$$

where

$$\kappa_1 = 1, \ \ \kappa_i = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i} \quad \textit{for} \ \ i > 1 \tag{6.40}$$

and,

$$\gamma_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n} \ . \tag{6.41}$$

Proof. To prove the theorem for the case $i = 1$ we start by expanding the vector y_i defined in the previous lemma in the eigenbasis $\{u_j\}$ as

$$y_1 = \sum_{j=2}^{n} \alpha_j u_j$$

where the α_j's are such that $\sum\limits_{j=2}^{n} |\alpha_j|^2 = 1$. From this we get,

$$\|p(A)y_1\|_2^2 = \sum_{j=2}^{n} |p(\lambda_j)\alpha_j|^2 \leq \max_{j=2,\ldots,n} |p(\lambda_j)|^2 \leq \max_{\lambda\in[\lambda_n,\lambda_2]} |p(\lambda)|^2 \ .$$

The result follows by a direct use of theorem 4.8 stated in Chapter IV. For the general case ($i \neq 1$), we can use the upper bound obtained by restricting the polynomials to be of the form

$$p(\lambda) = \frac{(\lambda_1 - \lambda)\cdots(\lambda_{i-1} - \lambda)}{(\lambda_1 - \lambda_i)\cdots(\lambda_{i-1} - \lambda_i)} q(\lambda)$$

where q is now any polynomial of degree $k - i$ such that $q(\lambda_i) = 1$. Proceeding as for the case $i = 1$, we arrive at the inequality,

$$\|p(A)y_i\|_2 \ \ \leq \ \ \max_{\lambda\in[\lambda_{i+1},\lambda_n]} \left| \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda}{\lambda_j - \lambda_i} q(\lambda) \right|$$

$$\leq \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i} \max_{\lambda \in [\lambda_{i+1}, \lambda_n]} |q(\lambda)| \ .$$

The result follows by minimizing this expression over all polynomials q satisfying thr constraint $q(\lambda_i) = 1$. ∎

6.2. Convergence of the Eigenvalues

We now turn our attention to the approximate eigenvalues. The following error bounds concerning the approximate eigenvalues $\lambda_i^{(m)}$ actually show that these converge to the corresponding eigenvalues of A if exact arithmetic were used.

Theorem 6.4 *The difference between the $i-th$ exact and approximate eigenvalues λ_i and $\lambda_i^{(m)}$ satisfies the double inequality,*

$$0 \leq \lambda_i - \lambda_i^{(m)} \leq (\lambda_1 - \lambda_n) \left(\frac{\kappa_i^{(m)} \tan \theta(v_1, u_i)}{C_{m-i}(1 + 2\gamma_i)} \right)^2 \tag{6.42}$$

where γ_i is defined in the previous theorem and $\kappa_i^{(m)}$ is given by

$$\kappa_1^{(m)} \equiv 1, \quad \kappa_i^{(m)} = \prod_{j=1}^{i-1} \frac{\lambda_j^{(m)} - \lambda_n}{\lambda_j^{(m)} - \lambda_i}, \quad i > 1 \ .$$

Proof. We prove the result only for the case $i = 1$. The first inequality is one of the properties proved for general projection methods when applied to Hermitian matrices. For the second, we note that

$$\lambda_1^{(m)} = \max_{x \neq 0, x \in \mathcal{K}_{m-1}} (Ax, x)/(x, x)$$

and hence,

$$\lambda_1 - \lambda_1^{(m)} = \min_{x \neq 0 \in \mathcal{K}_{m-1}} ((\lambda_1 I - A)x, x)/(x, x) \ .$$

Remembering that $\mathcal{K}_{m-1}$ is the set of all vectors of the form $q(A)v_1$ where q runs in the space $\mathbb{P}_{m-1}$ of polynomials of degree not exceeding $m-1$ this becomes

$$\lambda_1 - \lambda_1^{(m)} = \min_{0\neq q\in \mathbb{P}_{m-1}} \frac{((\lambda_1 - A)q(A)v_1, q(A)v_1)}{(q(A)v_1, q(A)v_1)} \ . \tag{6.43}$$

Expanding the initial vector v_1 in an orthonormal eigenbasis $\{u_j\}$ as

$$v_1 = \sum_{j=1}^{n} \alpha_j u_j$$

we find that

$$\lambda_1 - \lambda_1^{(m)} = \min_{0\neq q\in \mathbb{P}_{m-1}} \frac{\sum\limits_{j=2}^{n} (\lambda_1 - \lambda_j)|\alpha_j q(\lambda_j)|^2}{\sum\limits_{j=1}^{n} |\alpha_j q(\lambda_j)|^2}$$

from which we obtain the upper bound

$$\begin{aligned}
\lambda_1 - \lambda_1^{(m)} \quad & \leq (\lambda_1 - \lambda_n) \min_{0\neq q\in \mathbb{P}_{m-1}} \frac{\sum_{j=2}^{n} |\alpha_j q(\lambda_j)|^2}{\sum_{j=1}^{n} |\alpha_j q(\lambda_j)|^2} \\
& \leq (\lambda_1 - \lambda_n) \min_{0\neq q\in \mathbb{P}_{m-1}} \frac{\sum\limits_{j=2}^{n} |\alpha_j q(\lambda_j)|^2}{|\alpha_1 q(\lambda_1)|^2} \\
& \leq (\lambda_1 - \lambda_n) \min_{0\neq q\in \mathbb{P}_{m-1}} \max_{j=2,3,\ldots,n} \frac{|q(\lambda_j)|^2}{|q(\lambda_1)|^2} \frac{\sum\limits_{j=2}^{n} |\alpha_j|^2}{|\alpha_1|^2}
\end{aligned}$$

Defining $p(\lambda) = q(\lambda)/q(\lambda_1)$ and observing that the set of all p's when q runs in the space $\mathbb{P}_{m-1}$ is the set of all polynomials of degree not exceeding $m-1$ satisfying the constraint $p(\lambda_1) = 1$, we obtain

$$\lambda_1 - \lambda_1^{(m)} \leq (\lambda_1 - \lambda_n) \min_{p\in \mathbb{P}_{m-1}, p(\lambda_1)=1} \ \max_{\lambda\in[\lambda_n,\lambda_2]} |p(\lambda)|^2 \tan^2 \theta(u_1, v_1) \ .$$

The result follows by expressing the min-max quantity in the above expression using Chebyshev polynomials according to Theorem 4.8.

The general case $i > 1$ can be proved by using the Courant-Fisher characterization of $\lambda_i^{(m)}$. The i-th eigenvalue is the maximum of the Rayleigh quotient over the subspace of $\mathcal{K}_m$ that is orthogonal to the first $i-1$ approximate eigenvectors. This subspace can be shown to be the same as the subspace of all vectors of the form $q(A)v_1$ where q is a polynomial of degree not exceeding $m-1$ such that $q(\lambda_1^{(m)}) = q(\lambda_2^{(m)}) = \cdots = q(\lambda_{i-1}^{(m)}) = 0$. ∎

6.3. Convergence of the Eigenvectors

To get a bound for the angle between the exact and approximate eigenvectors produced by the Lanczos algorithm, we exploit the general result of Theorem 4.6 seen in Chapter IV. The theorem tells us that for any eigenpair λ_i, u_i of A there is an approximate eigenpair $\tilde{\lambda}, \tilde{u}_i$ such that,

$$\sin\left[\theta(u_i, \tilde{u}_i)\right] \leq \sqrt{1 + \frac{\gamma^2}{\delta_i^2}} \sin\left[\theta(u_i, \mathcal{K}_m)\right] \tag{6.44}$$

were δ_i is the distance between λ_i and the set of approximate eigenvalues other than $\tilde{\lambda}_i$ and $\gamma = \|\mathcal{P}_m A(I - \mathcal{P}_m)\|_2$. We notice that in the present situation we have

$$\begin{aligned}
(I - \mathcal{P}_m)A\mathcal{P}_m &= (I - V_m V_m^H)AV_m V_m^H \\
&= (I - V_m V_m^H)(V_m H_m + \beta_{m+1} v_{m+1} e_m^H)V_m^H \\
&= \beta_{m+1} v_{m+1} v_m^H ,
\end{aligned}$$

in which we used the relation (6.8). As a result

$$\gamma = \|\mathcal{P}_m A(I - \mathcal{P}_m)\|_2 = \|(I - \mathcal{P}_m)A\mathcal{P}_m\|_2 = \beta_{m+1} .$$

Since the angle between u_i and the Krylov subspace has been majorized in Theorem 6.3, a bound on the angle $\theta(u_i, \tilde{u}_i)$ can be readily obtained by combining these two results. For example, we can write

$$\begin{aligned}\sin\left[\theta(u_i,\tilde{u}_i)\right] &\le \sqrt{1+\beta_{m+1}^2/\delta_i^2}\ \sin\left[\theta(u_i,\mathcal{K}_m)\right]\\ &\le \sqrt{1+\beta_{m+1}^2/\delta_i^2}\ \tan\left[\theta(u_i,\mathcal{K}_m)\right]\\ &\le \frac{\kappa_i\sqrt{1+\beta_{m+1}^2\ /\ \delta_i^2}}{C_{m-i}(1+2\gamma_i)}\ \tan\theta(v_1,u_i)\end{aligned}$$

where the constants κ_i and γ_i are defined in Theorem 6.3.

7. Convergence of the Arnoldi Process

In this section we will analyze the speed of convergence of an approximate eigenvalue/ eigenvector obtained by Arnoldi's method to the exact pair. This will be done by considering the distance of a particular eigenvector u_i from the subspace $\mathcal{K}_m$. We will assume for simplicity that A is diagonalizable and define

$$\epsilon_i^{(m)} \equiv \min_{p\in \mathbb{P}^*_{m-1}} \ \max_{\lambda\in\sigma(A)-\lambda_i} |p(\lambda)|, \tag{6.45}$$

where $\mathbb{P}^*_{m-1}$ represents the set of all polynomials of degree not exceeding $m-1$ such that $p(\lambda_i)=1$. The following lemma relates the distance $\|(I-\mathcal{P}_m)u_i\|_2$ to the above quantity.

Lemma 6.2 *Assume that A is diagonalizable and that the initial vector v_1 in Arnoldi's method has the expansion $v_1=\sum_{k=1}^{k=n}\alpha_k u_k$ with respect to the eigenbasis $\{u_k\}_{k=1,\ldots,n}$ in which $\|u_k\|_2=1, k=1,2,\ldots,n$ and $\alpha_i\neq 0$. Then the following inequality holds:*

$$\|(I-\mathcal{P}_m)u_i\|_2 \le \xi_i\epsilon_i^{(m)}$$

where

$$\xi_i = \sum_{\substack{k=1\\ k\neq i}}^{n} \frac{|\alpha_j|}{|\alpha_i|}.$$

Proof. From the relation between $\mathcal{K}_m$ and $\mathbb{P}_{m-1}$ we have

$$\begin{aligned}
\|(I-\mathcal{P}_m)\alpha_i u_i\|_2 &= \min_{q\in\,\mathbb{P}_{m-1}} \|\alpha_i u_i - q(A)v_1\|_2 \\
&\leq \min_{q\in\,\mathbb{P}_{m-1},\ q(\lambda_i)=1} \|\alpha_i u_i - q(A)v_1\|_2,
\end{aligned}$$

and therefore, calling p the polynomial realizing the minimum on the right-hand-side

$$\|(I-\mathcal{P}_m)\alpha_i u_i\|_2 \leq \| \sum_{\substack{j=1\\ j\neq i}}^{n} \alpha_j p(\lambda_j)u_j\|_2 \leq \max_{j\neq i}|p(\lambda_j)| \sum_{\substack{j=1\\ j\neq i}}^{n} |\alpha_j|$$

which follows by using the triangle inequality and the fact that the component in the eigenvector u_1 is zero. The result is then established by dividing both members by $|\alpha_i|$. ∎

The question has been therefore converted into that of estimating the quantity (6.45) on which we will now focus. The quantity $\epsilon_i^{(m)}$ represents the smallest possible infinity norm over the set $\sigma(A)$, of all polynomials of the form $1-(z-\lambda_1)s(z)$, with s of degree not exceeding $m-1$. We seek an exact expression for $\epsilon_i^{(m)}$ or, equivalently for the best uniform approximation of the function unity on the set $\sigma(A)$, by polynomials of degree $\leq m$, satisfying the constraint that they vanish at the point λ_1. Without loss of generality we will restrict ourselves to the case $i=1$, i.e., we are interested in $\epsilon_1^{(m)}$. We will need the following lemma from approximation theory see, for example Cheney [16]. We recall that a set of functions satisfy the Haar condition on the points $x_1, x_2, \ldots, x_k$ if any linear combination f of these functions vanishes when $f(x_i)=0, i=1,\ldots,k$.

Lemma 6.3 *Let $\tilde{q}$ be the best uniform approximation of a continuous function f by a set of m polynomials satisfying the Haar condition on a compact set σ consisting of at least $m+1$ points. Then there exist at least $m+1$ points $\lambda_0,\ldots,\lambda_m$ of σ such that* *the error* $\overline{e(z)} = f(z) - \tilde{q}(z)$ *reaches its maximum modulus at the* λ_j*'s , i.e., such that:*

$$|e(\lambda_j)| = \max_{z \in \sigma} |e(z)| \qquad j = 0, 1, \ldots, m$$

Such points are called critical points.

Recall that we denote by $\mathbb{P}_m^*$ the set of polynomials p of degree $\leq m$ such that $p(\lambda_1) = 1$. In our case the function f is the function unity $f(x) \equiv 1$ and the set of polynomials by which it is approximated is the set of polynomials of degree $\leq m$, satisfying the constraint that they vanish at the point λ_1. This set is nothing but the set of polynomials $-1 + \mathbb{P}_m^*$ which constitutes a vector space of polynomials, of dimension m. Let $\lambda_2, \ldots, \lambda_{m+2}$ be the critical points corresponding to this best approximation as defined by the lemma. Then a useful *basis* of this space of polynomials is the basis consisting of the polynomials

$$\omega_j(z) = (z - \lambda_1)\hat{l}_j(z), \quad j = 2, \ldots, m+1, \tag{6.46}$$

where $\hat{l}_j$ is the Lagrange polynomial of degree $j - 1$,

$$\hat{l}_j(z) = \prod_{\substack{k=2 \\ k \neq j}}^{m+1} \frac{z - \lambda_k}{\lambda_j - \lambda_k}, \quad j = 2, \ldots, m+1. \tag{6.47}$$

With this we can prove the following lemma.

Lemma 6.4 *The underdetermined linear system of* m *equations and* $m + 1$ *unknowns* $z_i, i = 2, \ldots, m+2$

$$\sum_{i=2}^{m+2} \omega_j(\lambda_i) z_i = 0, \quad j = 2, 3, \ldots, m+1 \tag{6.48}$$

admits the nontrivial solution

$$z_i = \prod_{\substack{k=2 \\ k \neq i}}^{m+2} \frac{\lambda_1 - \lambda_k}{\lambda_i - \lambda_k}, \quad i = 2, \ldots, m+2.$$

Proof. Because of the Haar condition, the system of polynomials $\{\omega_j\}_{j=2,\ldots,m+1}$, forms a basis and therefore there exists a nontrivial solution to the above linear system. By the definition of the Lagrange polynomials, all the terms in the $i-$th equation vanish except those corresponding to $j = i$ and to $j = m+2$. Thus, the i^{th} equation can be rewritten as

$$(\lambda_i - \lambda_1)z_i + z_{m+2}(\lambda_{m+2} - \lambda_1) \prod_{\substack{k=2 \\ k \neq i}}^{m+1} \frac{\lambda_{m+2} - \lambda_k}{\lambda_i - \lambda_k} = 0.$$

The unknown z_{m+2} can be assigned an arbitrary nonzero value (since the system is underdetermined) and then the other unknowns are determined uniquely by:

$$\frac{z_i}{z_{m+2}} = -\frac{(\lambda_{m+2} - \lambda_1)}{(\lambda_i - \lambda_1)} \prod_{\substack{k=2, \\ k \neq i}}^{m+1} \frac{\lambda_{m+2} - \lambda_k}{\lambda_i - \lambda_k} = -\prod_{\substack{k=1 \\ k \neq i}}^{m+1} \frac{(\lambda_{m+2} - \lambda_k)}{(\lambda_i - \lambda_k)}.$$

Multiplying numerator and denominator by $(\lambda_i - \lambda_{m+2})$ we get

$$z_i = \frac{C}{\lambda_1 - \lambda_i} \prod_{\substack{k=2 \\ k \neq i}}^{m+2} \frac{1}{\lambda_i - \lambda_k}$$

where C is the following constant, which depends on the choice of z_{m+2},

$$C \equiv z_{m+2} \prod_{k=1}^{m+2} (\lambda_{m+2} - \lambda_k) \ .$$

The result follows by choosing z_{m+2} so that,

$$C = \prod_{k=2}^{m+2} (\lambda_1 - \lambda_k).$$

∎

We should point out that the solution $\{z_k\}$ *does not depend on the basis* chosen for the space of polynomials $-1 + \mathbb{P}_m^*$. For example choosing the usual power basis $(z-\lambda_1)z^{j-1}, j = 1, \ldots, m$, yields the same set $\{z_j\}$; see Exercise P-6.12. The basis $\{\omega_i\}$ is far more convenient than the power basis for determining this solution because of the simplicity of the resulting linear system (6.48). The following lemma will now be proved.

Lemma 6.5 *Let $\bar{p}$ be the (unique) polynomial of degree m satisfying the constraint $p(\lambda_1) = 1$, and having the smallest infinity norm on a compact set σ consisting of at least $m + 1$ points. Let the $m + 1$ critical points as defined by Lemma 6.3 be labeled $\lambda_2, \ldots, \lambda_{m+2}$. Let $z_k, k = 2, \ldots, m+2$ be any solution of the linear system (6.48) and write each z_k in the form $z_k = \delta_k e^{-i\theta_k}$ where δ_k is real and positive and θ is real. Then, $\bar{p}$ can be expressed as*

$$\bar{p}(z) = \frac{\sum_{k=2}^{m+2} e^{i\theta_k} l_k(z)}{\sum_{k=2}^{m+2} e^{i\theta_k} l_k(\lambda_1)} \tag{6.49}$$

where l_k is the Lagrange polynomial of degree m

$$l_k(z) = \prod_{\substack{j=2 \\ j\neq k}}^{m+2} \frac{z - \lambda_j}{\lambda_k - \lambda_j}.$$

Proof. By the equations (6.48) that define the z_k's we have for any v belonging to the space of polynomials $-1 + \mathbb{P}_m^* = \mathrm{span}\{\omega_i\}_{i=2,\ldots,m+1}$,

$$\sum_{k=2}^{m+2} \delta_k e^{-i\theta_k} v(\lambda_k) = 0. \tag{6.50}$$

Let $\bar{p}$ the polynomial defined by (6.49). We must show that

$$\|\bar{p} + v\|_\infty \geq \|\bar{p}\|_\infty \tag{6.51}$$

for any v in $-1 + \mathbb{P}^*_m$, where $\|.\|_\infty$ represents the infinity norm over the set σ. Let us set

$$\rho = \left[\sum_{k=2}^{m+2} e^{i\theta_k} l_k(\lambda_1) \right]^{-1} . \tag{6.52}$$

Notice that $|\rho|$ is the uniform norm of $\bar{p}$ in σ. From (6.50) it is clear that for some k' we have

$$\Re e \; \left[\rho e^{-i\theta_{k'}} v(\lambda_{k'}) \right] \geq 0.$$

Therefore,

$$\begin{aligned} \|\bar{p} + v\|^2_\infty &= \max_{j=2,\ldots,m+2} |\bar{p}(\lambda_j) + v(\lambda_j)|^2 \\ &\geq |\bar{p}(\lambda_{k'}) + v(\lambda_{k'})|^2 \\ &\geq |\rho e^{-i\theta_{k'}} + v(\lambda_{k'})|^2 \\ &= |\rho|^2 + |v(\lambda_{k'})|^2 + 2\, \Re e \,\{\rho e^{-i\theta_{k'}} v(\lambda_{k'})\} \\ &\geq |\rho|^2 = \|\bar{p}\|^2_\infty \end{aligned}$$

which shows that (6.51) is true and completes the proof. ∎

We are now ready to state the main result of this section.

Theorem 6.5 *Let $m < n$. Then there exist m eigenvalues of A which can be labeled $\lambda_2, \lambda_3, \lambda_{m+1}$ such that:*

$$\epsilon_1^{(m)} = \left(\sum_{j=2}^{m+1} \prod_{k=2,k\neq j}^{m+1} \frac{|\lambda_k - \lambda_1|}{|\lambda_k - \lambda_j|} \right)^{-1} . \tag{6.53}$$

Proof. Observe that the solution of the linear system (6.48) satisfies $z_j = l_j(\lambda_1)$. The proof is obtained by simply replacing this solution in the expression of ρ defined in (6.52). Note that the polynomials for the lemmas are of degree m whereas the the candidate polynomials in (6.45) are of degree $m - 1$. ∎

For the case where the eigenvalue is in the outermost part of the spectrum, the above expression can be interpreted as follows. In general, the distances $|\lambda_k - \lambda_1|$ are larger than the corresponding distances $|\lambda_k - \lambda_j|$ of the denominator. This is illustrated in Figure (6.2). Therefore, many of the products will be large when m is large and the inverse of their sum will be small. This is independent of the actual locations of the critical points which are not known. The conclusion is that the eigenvalues that are in the outermost part of the spectrum are likely to be well approximated.

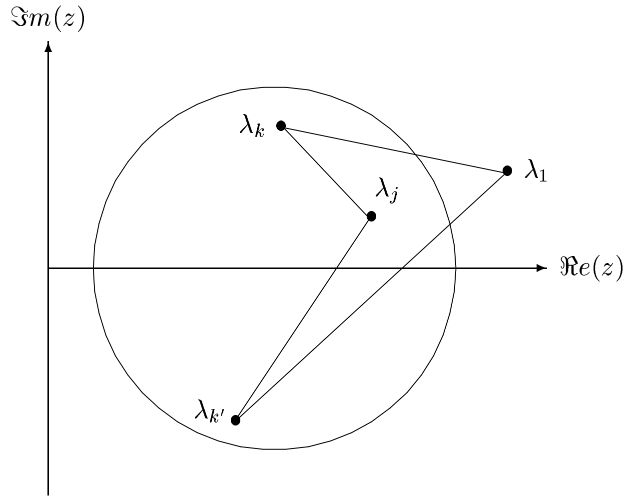

Figure 6.2 Illustration of Theorem 6.5 for λ_1 in the outermost part of the spectrum of A.

We can illustrate the above theorem with a few examples.

Example 6.4 Assume that

$$\lambda_k = \frac{k-1}{n-1}, \quad k = 1, 2, \ldots n,$$

and consider the special case when $m = n - 1$. Then,

$$\epsilon_1^{(m)} = \frac{1}{2^m - 1}.$$

Indeed, since $m = n-1$ there is no choice for the λ_j's in the theorem but to be the remaining eigenvalues and (6.53) yields,

$$\begin{aligned}
(\epsilon_1^{(m)})^{-1} &= \sum_{j=2}^{m+1} \prod_{\substack{k=2\\k\neq j}}^{m+1} \frac{|k-1|}{|k-j|} \\
&= \sum_{j=2}^{m+1} \frac{m!}{(j-1)!(m+j-1)!} \\
&= \sum_{j=1}^{m} \binom{j}{m} = 2^m - 1 \ .
\end{aligned}$$

Example 6.5 Consider now a uniform distribution of eigenvalues over a circle instead of a real line,

$$\lambda_k = e^{i\frac{2(k-1)\pi}{n}}, \ k = 1, 2, \ldots, n \ .$$

We assume once more that $m = n - 1$. Then we have

$$\epsilon_1^{(m)} = \frac{1}{m} \ .$$

To prove the above formula, we utilize again the fact that the eigenvalues involved in the theorem are known to be $\lambda_2, \lambda_3, ..., \lambda_n$. We define $\omega = e^{2i\pi/n}$ and write each product term in the formula (6.53) as

$$\begin{aligned}
\prod_{\substack{k=2\\k\neq j}}^{m+1} \frac{|\omega^{k-1} - 1|}{|\omega^{k-1} - \omega^{j-1}|} &= \prod_{\substack{k=1\\k\neq j}}^{m} \frac{|\omega^{k} - 1|}{|\omega^{k} - \omega^{j}|} \\
&= \left[\prod_{k=1}^{m} |\omega^k - 1|\right] \left[|1-\omega^j| \prod_{\substack{k=1\\k\neq j}}^{m} |\omega^k - \omega^j|\right]^{-1} \ .
\end{aligned}$$

Recalling that the ω^k's are the powers of the n-th root of unity, we observe that a simple renumbering of the products in the denominator will show that the numerator and denominator have the same modulus. Hence the above product term is equal to one and by summing these products and inverting, we will get the desired result.

The above two examples show a striking difference in behavior between two seemingly similar situations. The complex uniform distribution of eigenvalues over a circle is a much worse situation than that of the uniform distribution over a line segment. It indicates that there are cases where the eigenvalues will converge extremely slowly. Note that this poor convergence scenario may even occur if the matrix A is normal, since it is only the distribution of the eigenvalues that cause the difficulty.

Apart from the qualitative interpretation given above, it is also possible to give a simple explicit upper bounds for $\epsilon_i^{(m)}$.

Proposition 6.10 *Let $C(c,\rho)$ be a circle of center c and radius ρ that encloses all the eigenvalues of A except λ_1. Then,*

$$\epsilon_1^{(m)} \leq \left(\frac{\rho}{|\lambda_1 - c|}\right)^{m-1} .$$

Proof. An upper bound is obtained by using the particular polynomial $q(z) = (z-c)^{m-1}/(\lambda_1 - c)^{m-1}$ from which we get

$$\epsilon_1^{(m)} \leq \max_{j=2,3,\ldots,n} \left(\frac{|\lambda_j - c|}{|\lambda_1 - c|}\right)^{m-1} \leq \rho^{m-1}/|\lambda_1 - c|^{m-1} .$$

■

It was seen in Chapter IV (Lemma 4.3) that the polynomial used in the proof is actually optimal.

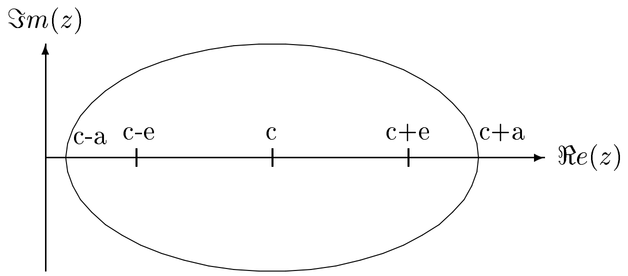

Figure 6.3 Ellipse containing the spectrum of A.

Still from what was seen on Chebyshev polynomials in Chapter IV. we may be able to get a better estimate of $\epsilon_1^{(m)}$ if we can enclose the eigenvalues of the matrix A in an ellipse centered at c with focal distance e and major semi-axis a, as is illustrated in Figure 6.3. In this situation the results on the Chebyshev polynomials of the first kind allow us to state the following theorem.

Theorem 6.6 *Assume that all the eigenvalues of A expect λ_1 lie inside the ellipse centered at c, with foci $c+e, c-e$ and with major semi axis a. Then,*

$$\epsilon_1^{(m)} \leq \frac{C_{m-1}\left(\frac{a}{e}\right)}{|C_{m-1}\left(\frac{\lambda_1 - c}{e}\right)|} \tag{6.54}$$

where C_{m-1} is the Chebyshev polynomial of degree $m-1$ of the first kind. In addition, the relative difference between the left and the right hand sides tends to zero as m tends to infinity.

PROBLEMS

P-6.1 To measure the degree of invariance of a subspace X with respect to a matrix A, we define the measure $v(X, A) = \|(I-P)AP\|_2$ where P is the orthogonal projector onto the subspace. (1) Show that if X is invariant then $v(X, A) = 0$. (2) Show that when X is the m-th Krylov subspace generated from some initial vector v, then $v(X, A) = \beta_{m+1}$. (3) Let $r_i, i = 1, \ldots, m$ be the residual vectors associated with the approximate eigenvalues obtained from an orthogonal projection process onto X, and let $R = [r_1, ..., r_m]$. Show that $v(X, A) = \|R\|_2$.

P-6.2 Consider the matrix

$$A = \begin{pmatrix} 0 & & & & & 1 \\ 1 & & & & & 0 \\ & 1 & & & & 0 \\ & & 1 & & & \vdots \\ & & & \ddots & & \vdots \\ & & & & 1 & 0 \end{pmatrix}$$

(1) What are eigenvalues of A? (2) What is the m-th Krylov subspace associated with A when $v_1 = e_1$, the first column of the identity matrix? (3) What are the approximate eigenvalues obtained from Arnoldi's method in this case? How does this relate to Example 6.5?

P-6.3 Assume that k Schur vectors have already been computed and let P be an orthogonal projector associated with the corresponding invariant subspace. Assume that Arnoldi's method is applied to the matrix $(I-P)A$ starting with a vector that is orthogonal to the invariant subspace. Show that the Hessenberg matrix thus obtained is the same as the lower $(m-k) \times (m-k)$ principal submatrix obtained from an implicit deflation procedure. Show that an approximate Schur vector associated with the corresponding projection procedure is an approximate Schur vector for A. This suggests another implementation of the implicit deflation procedure seen in Section 2.3 in which only the $(m-k) \times (m-k)$ Hessenberg matrix is used. Give a corresponding new version of Algorithm 6.4. What are the advantages and disadvantages of this approach?

P-6.4 Show that for the Lanczos algorithm one has the inequality

$$\max_{i=1,2,\ldots,m} [\beta_{i+1}^2 + \alpha_i^2 + \beta_{i-1}^2]^{1/2} \leq \max_{j=1,\ldots,n} |\lambda_j|$$

Show a similar result in which max is replaced by min.

P-6.5 Consider a matrix A that is *skew-Hermitian*. (1) Show that the eigenvalues of A are purely imaginary. What additional property do they satisfy in the particular case when A is *real* skew-symmetric? [Hint: eigenvalues of real matrices come in complex conjugate pairs...] What can you say of a real skew-symmetric matrix of *odd* dimension n? (2) Assume that Arnoldi's procedure is applied to A starting with some arbitrary vector v_1. Show that the algorithm will produce scalars h_{ij} such that

$$\begin{aligned} & h_{ij} = 0, \quad \text{for} \quad i < j-1 \\ & \Re e[h_{jj}] = 0, j = 1, 2, ..., m \\ & h_{j,j+1} = -h_{j+1,j} j = 1, 2, ..., m \end{aligned}$$

(3) From the previous result show that in the particular where A is real skew-symmetric and v_1 is real, then the Arnoldi vectors v_j satisfy

a two term recurrence of the form

$$\beta_{j+1}v_{j+1} = Av_j + \beta_j v_{j-1}$$

(4) Show that the approximate eigenvalues of A obtained from the Arnoldi process are also purely imaginary. How do the error bounds of the Lanczos algorithm (Hermitian case) extend to this case?

P-6.6 How do the results of the previous problem extend to the case where $A = \alpha I + S$ where α is a real scalar and S is skew-Hermitian or skew symmetric real?

P-6.7 We consider the following tridiagonal matrix A_n of size $n \times n$

$$A_n = \begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & . & & \\ & 1 & . & 1 & \\ & & . & 2 & 1 \\ & & & 1 & 2 \end{pmatrix}.$$

(1) Consider the vector z of length n whose $j-th$ component is $\sin j\theta$ where θ is a real parameter such that $0 < \theta \leq \pi/2$. Show that

$$(2(1+\cos\theta)I - A_n)z = \sin((n+1)\theta)e_n$$

where $e_n = (0,0,...0,1)^H$. (2) Using the previous question find all the eigenvalues and corresponding eigenvectors of A_n. (3) Assume now that m steps of the Lanczos algorithm are performed on A_n with the starting vector $v_1 = e_1 = (1,0,...,0)^H$. (3.a) Show that the Lanczos vectors v_j are such that $v_j = e_j$, $j = 1,2,...,m$. (3.b) What is the matrix T_m obtained from the Lanczos procedure? What are the approximate eigenvalues and eigenvectors? (Label all the eigenvalues in decreasing order). (3.c) What is the residual vector and the residual norm associated with the first approximate eigenvalue $\lambda_1^{(m)}$? [Hint: It will be admitted that

$$\sin^2\frac{\pi}{(m+1)} + \sin^2\frac{2\pi}{(m+1)} + ... + \sin^2\frac{m\pi}{(m+1)} = \frac{m+1}{2}]$$

How would you explain the fact that convergence is much slower than expected?

P-6.8 Show that the vector v_{m+1} obtained at the last step of Arnoldi's method is of the form $v_{m+1} = \gamma p_m(A)v_1$, in which γ is a certain normalizing scalar and p_m is the characteristic polynomial of the Hessenberg matrix H_m.

P-6.9 Develop a modified version of the non-Hermitian Lanczos algorithm that produces a sequence of vectors v_i, w_i that are such that each v_i is orthogonal to every w_j with $j \neq i$ and $\|v_i\|_2 = \|w_i\|_2 = 1$ for all i. What does the projected problem become?

P-6.10 Develop a version of the non-Hermitian Lanczos algorithm that produces a sequence of vectors v_i, w_i which satisfy

$$(v_i, v_j) = \pm\delta_{ij},$$

but such that the matrix T_m is Hermitian tridiagonal. What does the projected problem become in this situation? How can this version be combined with the version defined in the previous exercise?

P-6.11 Using the notation of Section 3.2 prove that $q_{j+k}(x) = x^k p_j(x)$ is orthogonal to the polynomials $p_1, p_2, \ldots, p_{j-k}$, assuming that $k \leq j$. Show that if we orthogonalized q_{j+k} against $p_1, p_2, \ldots, p_{j-k}$, we would obtain a polynomial that is orthogonal to all polynomials of degree $<j + k$. Derive a general look-ahead non-Hermitian Lanczos procedure based on this observation.

P-6.12 It was stated after the proof of Lemma (6.4) that the solution of the linear system (6.48) is independent of the basis chosen to establish the result in the proof of the lemma. 1) Prove that this is the case. 2) Compute the solution directly using the power basis, and exploiting Vandermonde determinants.

NOTES AND REFERENCES. There has been several papers published on Arnoldi's method and its variants for solving eigenproblems. The original paper by Arnoldi [2] came out about one year after Lanczos' breakthrough paper [89] and is quite different in nature. The author hints that his method can be viewed as a projection method and that it might be used to approximate eigenvalues of large matrices. Note that the primary goal of the method is to reduce an arbitrary (dense) matrix to Hessenberg form. At the time, the QR algorithm was not yet invented, so the Hessenberg form was desired only because it leads to a simlpe recurrence for the characteristic polynomial. The 1980 paper by Saad [139] showed that the method could indeed be quite useful as a projection method to solve large eigenproblems, and gave a few variations of it. Later, sophisticated versions have been developed and used in realistic applications, see [17, 105, 106, 115, 123, 154], among others. During roughly the same period, much work was devoted to exploiting the basic non-Hermitian Lanczos algorithm by Parlett and co-workers [125] and by Cullum and Willoughby [24, 26] and Cullum, Kerner and Willoughby [22]. The first successful application of the code in a real life problem seems to be in the work by Carnoy and Geradin [12] who used a version of the algorithm in a finite element model.

The block Lanczos algorithm seems to have been developed first by Golub and Underwood [61]. The equivalent Block Arnoldi algorithm, has not been given much attention, except in control problems where it is closely associated with the notion of controllability for the multiple-input case [6]. In fact Arnoldi's method (single input case) and its block analogue (multiple input case) are useful in many areas in control, see for example [149, 150].

The error bounds on the Hermitian Lanczos algorithm are from [138]. Bounds of a different type have been proposed by Kaniel [83] (however there were a few errors for the case $i>1$ in Kaniel's original paper and some of these errors were later corrected by Paige [112]). We have omitted to discuss similar bounds for the Block Lanczos algorithm but these were also developed in Saad [138]. The convergence theory for the Arnoldi process is adapted from Saad [141].

The various implementations of the Lanczos algorithm in the Hermitian case are covered in detail in Parlett's book [118]. Implementations on massively parallel machines have recently been discussed by Petiton [126] on the CM-2 and by Scott [161] on the iPSC/2.

Concerning software, there is little that is publically available. Cullum and Willoughby offer a FORTRAN code for the Hermitian case in their book [25] based on the Lanczos algorithm without any form of reorthogonalization. A similar (research) code was also developed by Parlett and Reid [122]. Recently, Freund, Gutknecht, and Nachtigal published a FORTRAN implementation of their Look-Ahead Lanczos algorithm [49]. We know of no other codes based on the Lanczos algorithm with or without reorthogonalization. There has been a few implementations of the Hermitian Lanczos and the Block Lanczos algorithm with some form of reorthogonalization. We refer to the survey by Parlett concerning software availability in 1984 [119]. Interestingly enough, there has been very little new happening in the software scene since then, so this survey seems almost up to date, in 1991. ♠

Chapter VII

Acceleration Techniques and Hybrid Methods

Many of the early algorithms for eigenvalue extraction were based on using the powers of the matrix A. The prototype of these techniques is the power method, a technique that is attractive because of its simplicity but whose convergence rate may be unacceptably slow. Acceleration methods can be valuable tools for speeding up the convergence of this and other algorithms. In this chapter we will present a number of techniques that are commonly termed polynomial acceleration techniques for vector iterations. They are based on an interesting blend of approximation theory and numerical linear algebra. A polynomial iteration takes the form $z_k = p_k(A)z_0$ where p_k is a polynomial which is determined from some knowledge on the distribution of the eigenvalues of A. A fundamental problem, which will utilize ideas from approximation theory, lies in computing a good polynomial p_k. By combining a basic method such as Arnoldi's method, with polynomial acceleration, efficient algorithms for computing an eigenvector or a few eigenvectors of a large sparse matrix can be derived.

1. The Basic Chebyshev Iteration

Let A be a real nonsymmetric (or non Hermitian complex) matrix of dimension n and consider the eigenvalue problem,

$$Au = \lambda u. \tag{7.1}$$

Let $\lambda_1, \cdots, \lambda_n$ be the eigenvalues of A labeled in decreasing order of their real parts, and suppose that we are interested in λ_1 which is initially assumed to be real.

We consider a polynomial iteration of the form: $z_k = p_k(A)z_0$, where z_0 is some initial vector and where p_k is a polynomial of degree k. We would like to choose p_k in such a way that the vector z_k converges rapidly towards an eigenvector of A associated with λ_1 as k tends to infinity. Assuming for simplicity that A is diagonalizable, we expand z_0 in the eigenbasis $\{u_i\}$ as,

$$z_0 = \sum_{i=1}^{n} \theta_i u_i,$$

which leads to the following expression for $z_k = p_k(A)z_0$:

$$z_k = \sum_{i=1}^{n} \theta_i p_k(\lambda_i) u_i = \theta_1 p_k(\lambda_1) u_1 + \sum_{i=2}^{n} \theta_i p_k(\lambda_i) u_i. \tag{7.2}$$

The above expansion shows that if z_k is to be a good approximation of the eigenvector u_1, then the second term must be much smaller that the first and this can be achieved by making every $p_k(\lambda_j)$, with $j \neq 1$, small in comparison with $p_k(\lambda_1)$. This leads us to seek a polynomial which takes 'small' values on the discrete set

$$R = \{\lambda_2, \lambda_3, \cdots, \lambda_n\},$$

and which satisfies the normalization condition

$$p_k(\lambda_1) = 1. \tag{7.3}$$

An ideal such polynomial would be one which minimizes the (discrete) uniform norm on the discrete set R over all polynomials of degree k satisfying (7.3). However, this polynomial is impossible to compute without the knowledge of all eigenvalues of A and as a result this approach has little practical value. A simple and common alternative, is to replace the discrete min-max polynomial by the continuous one on a domain containing R but excluding λ_1. Let E be such a domain in the complex plane, and let $\mathbb{P}_k$ denote the space of all polynomials of degree not exceeding k. We are thus seeking a polynomial p_k which achieves the minimum

$$\min_{p\in \mathbb{P}_k,\ p(\lambda_1)=1} \quad \max_{\lambda\in E} |p(\lambda)|. \tag{7.4}$$

For an arbitrary domain E, it is difficult to solve explicitly the above min-max problem. Iterative methods can be used, however, and the exploitation of the resulting min-max polynomials for solving eigenvalue problems constitutes a promising research area. A preferred alternative is to restrict E to be an ellipse having its center on the real line, and containing the unwanted eigenvalues $\lambda_i, i = 2, \cdots, n$.

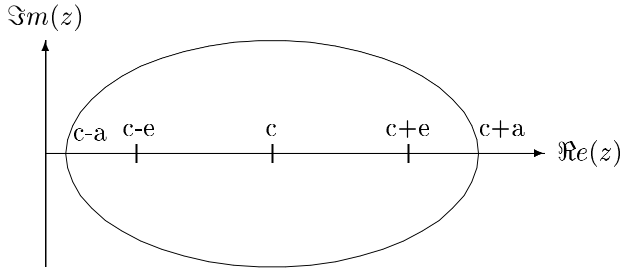

Figure 7.1 Ellipse containing the spectrum of A with e real.

Let $E(c, e, a)$ be an ellipse containing the set

$$R = \{\lambda_2, \lambda_3, \cdots, \lambda_n\},$$

and having (real) center c, foci $c + e$, $c - e$, and major semi-axis a. When A is real the spectrum of A is symmetric with respect to the real axis, so we can restrict $E(c, e, a)$ to being symmetric as well. In other words, the main axis of the ellipse is either the real axis or a line parallel to the imaginary axis. Therefore, a and e are either both real or both purely imaginary. These two cases are illustrated in Figure 7.1 and Figure 7.2 respectively.

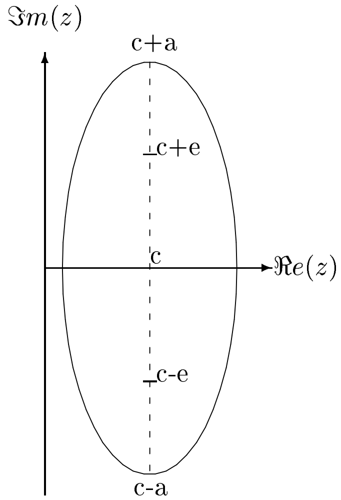

Figure 7.2 Ellipse containing the spectrum of A, with e purely imaginary.

A result that is known in approximation theory and shown in Section IV-4 is that when E is the ellipse $E(c, e, a)$ in (7.4), an asymptotically best min-max polynomial is the polynomial

$$p_k(\lambda) = \frac{C_k[(\lambda - c)/e]}{C_k[(\lambda_1 - c)/e]} , \tag{7.5}$$

where C_k is the Chebyshev polynomial of degree k of the first kind.

The computation of $z_k = p_k(A)z_0, k = 1, 2, \cdots$, is simplified by the three–term recurrence for the Chebyshev polynomials,

$$C_1(\lambda) = \lambda, \qquad C_0(\lambda) = 1,$$
$$C_{k+1}(\lambda) = 2\lambda C_k(\lambda) - C_{k-1}(\lambda), \qquad k = 1, 2, \cdots \ .$$

Letting $\rho_k = C_k[(\lambda_1 - c)/e], k = 0, 1, \cdots$, we obtain

$$\rho_{k+1}p_{k+1}(\lambda) = C_{k+1}[\frac{\lambda - c}{e}] = 2\ \frac{\lambda - c}{e}\rho_k p_k(\lambda) - \rho_{k-1}p_{k-1}(\lambda).$$

We can simplify this further by defining $\sigma_{k+1} \equiv \rho_k/\rho_{k+1}$,

$$p_{k+1}(\lambda) = 2\sigma_{k+1}\frac{\lambda - c}{e}p_k(\lambda) - \sigma_k\sigma_{k+1}p_{k+1}(\lambda).$$

A straightforward manipulation using the definitions of σ_i, ρ_i and the three–term recurrence relation of the Chebyshev polynomials shows that $\sigma_i, i = 1, 2, \cdots$, can be obtained from the recurrence,

$$\sigma_1 = \frac{e}{\lambda_1 - c};$$
$$\sigma_{k+1} = \frac{1}{\frac{2}{\sigma_1} - \sigma_k}, \quad k = 1, 2, \cdots \quad .$$

The above two recursions defining z_k and σ_k can now be assembled together to yield a basic algorithm for computing $z_k = p_k(A)z_0,\ k \geq 1$. Although λ_1 is not known, recall that it is used in the denominator of (7.5) for scaling purposes only, so we may replace it by some approximation ν in practice.

ALGORITHM **7.1 Chebyshev Iteration**

1. *Start: Choose an arbitrary initial vector* z_0 *and compute*

$$\sigma_1 = \frac{e}{\lambda_1 - c}, \tag{7.6}$$
$$z_1 = \frac{\sigma_1}{e}(A - cI)z_0. \tag{7.7}$$

2. *Iterate: For* $k = 1, 2, \cdots$, *until convergence do:*

$$\sigma_{k+1} = \frac{1}{2/\sigma_1 - \sigma_k}, \tag{7.8}$$
$$z_{k+1} = 2\frac{\sigma_{k+1}}{e}(A - cI)z_k - \sigma_k\sigma_{k+1}z_{k-1}\ . \tag{7.9}$$

An important detail, which we have not considered for the sake of clarity, concerns the case when e is purely imaginary. It can be shown quite easily that even in this situation the above recursion can still be carried out in real arithmetic. The reason for this is that the scalars σ_{k+1}/e and $\sigma_{k+1}\sigma_k$ in the above algorithm are real numbers. The primary reason for scaling by $C_k[(\lambda_1 - c)/e]$ in (7.5) is to avoid overflow but we have just given another reason, namely avoid complex arithmetic when e is purely imaginary.

1.1. Convergence Properties.

In order to understand the convergence properties of the sequence of approximations z_k we consider its expansion (7.2) and examine the behavior of each coefficient of u_i, for $i \neq 1$. By the definition of p_k we have:

$$p_k(\lambda_i) = \frac{C_k[(\lambda_i - c)/e]}{C_k[(\lambda_1 - c)/e]}.$$

From the standard definition of the Chebyshev polynomials in the complex plane seen in Chapter IV, the above expression can be rewritten as

$$p_k(\lambda_i) = \frac{w_i^k + w_i^{-k}}{w_1^k + w_1^{-k}}, \tag{7.10}$$

where w_i represents the root of largest modulus of the equation in w:

$$\frac{1}{2}(w + w^{-1}) = \frac{\lambda_i - c}{e}. \tag{7.11}$$

From (7.10), $p_k(\lambda_i)$ is asymptotic to $[w_i/w_1]^k$, hence the following definition.

Definition 7.1 *We will refer to $\kappa_i = |w_i/w_1|$ as the damping coefficient of λ_i relative to the parameters c, e. The convergence ratio $\tau(\lambda_1)$ of λ_1 is the largest damping coefficient κ_i for $i \neq 1$.*

The meaning of the definition is that each coefficient in the eigenvector u_i of the expansion (7.2) behaves like κ_i^k, as k tends to infinity. The damping coefficient $\kappa(\lambda)$ can obviously be also defined for any value λ in the complex plane, not necessarily an eigenvalue. Given a set of r wanted eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_r$, the definition 7.1 can be extended for an eigenvalue λ_j $j \le r$ as follows. The damping coefficient for any 'unwanted' eigenvalue $\lambda_i, i > r$ must simply be redefined as $|w_i/w_j|$ and the convergence ratio $\tau(\lambda_j)$ with respect to the given ellipse is the largest damping coefficient κ_l, for $l = r+1, \ldots, n$.

One of the most important features in Chebyshev iteration lies in the expression (7.11). There are infinitely many points λ in the complex plane whose damping coefficient $\kappa(\lambda)$ has the same value κ. These points λ are defined by $(\lambda - c)/e = (w + w^{-1})/2$ and $|w/w_1| = \kappa$ where κ is some constant, and belong to the same confocal ellipse $E(c, e, a(\kappa))$. Thus a great deal of simplification can be achieved by locating those points *that are real* as it is preferable to deal with real quantities than imaginary ones in the above expression defining κ_i. As was seen in Section IV-4 the mapping $J(w) = \frac{1}{2}(w + w^{-1})$, transforms a circle into an ellipse in the complex plane. More precisely, for $w = \rho e^{i\theta}$, $J(w)$ belongs to an ellipse of center the origin, focal distance 1, and major semi-axis $\rho = \frac{1}{2}(\rho + \rho^{-1})$. Moreover, given the major semi-axis α of the ellipse, the radius ρ is determined by $\rho = \frac{1}{2}[\alpha + (\alpha^2 - 1)^{1/2}]$. As a consequence the damping coefficient κ_i is simply ρ_i/ρ_1 where $\rho_i \equiv \frac{1}{2}[\alpha_i + (\alpha_i^2 - 1)^{1/2}]$ and α_i is the major semi-axis of the ellipse centered at the origin, with focal distance one and passing through $(\lambda_j - c)/e$. Since $\alpha_1 > \alpha_i, i = 2, 3, \cdots, n$, it is easy to see that $\rho_1 > \rho_i$, $i > 1$, and hence that the process will converge. Note that there is a further mapping between λ_j and $(\lambda_j - c)/e$ which transforms the ellipse $E(c, e, a_j)$ into the ellipse $E(0, 1, \alpha_j)$ where a_j and α_j are related by $\alpha_j = a_j/e$. Therefore, the above expression for the damping coefficient can be rewritten as:

$$\kappa_i = \frac{\rho_i}{\rho_1} = \frac{a_i + (a_i^2 - 1)^{1/2}}{a_1 + (a_1^2 - 1)^{1/2}}, \tag{7.12}$$

where a_i is the major semi-axis of the ellipse of center c, focal distance e, passing through λ_i. From the expansion (7.2), the vector z_k converges to $\theta_1 u_1$, and the error behaves like $\tau(\lambda_1)^k$.

The algorithm described above does not address a certain number of practical issues. For example, the parameters c and e will not generally be known beforehand, and their estimation is required. The estimation is typically done in a dynamic manner. In addition, the algorithm does not handle the computation of more than one eigenvalue. In particular what can we do in case λ_1 is complex, i.e., when λ_1 and $\lambda_2 = \bar{\lambda}_1$ form a complex pair?

2. Arnoldi–Chebyshev Iteration

As was just argued, Chebyshev iteration alone has a few important limitations. In fact it is rarely used as a single vector iteration in practice but rather combined with some other technique. The purpose of this section is to describe one such combination.

Suppose that $E(c, e, a)$ contains all the eigenvalues of A except for a few of them. Looking closely at the expansion of z_k, we observe that it will typically contain more than just an approximation to u_1. In general, the vector has the form

$$z_k = \theta_1 u_1 + \theta_{i_1} u_{i_1} + \ldots + \theta_{i_p} u_{i_p} + \epsilon, \tag{7.13}$$

where $\lambda_{i_1}, \cdots, \lambda_{i_p}$ are the eigenvalues outside $E(c, e, a)$ and ϵ is a relatively small term in comparison with the first r ones. All we need is a method to extract those eigenvalues from the single vector z_k. We will refer to such a method as a purification process. One process of this type can be achieved via the Arnoldi method seen in the preceding Chapter. In fact any of the projection techniques seen earlier can be used as well.

2.1. Purification by Arnoldi's Method

An important property of Arnoldi's method seen in Chapter VI, is that if the initial vector v_1 is exactly in an invariant subspace of dimension r and not in any invariant subspace of smaller dimension, i.e., if the grade of v_1 is r, then the algorithm stops at step

$m = r$, because we will obtain $\|\hat{v}_{r+1}\| = 0$. However, as Proposition 6.2 shows in this case K_r will be invariant, which implies by Proposition 4.3 that the r computed eigenvalues are exact.

This suggests that a good choice for the initial vector v_1 in Arnoldi's method would be to take a vector which is close to being in an invariant subspace of small dimension. Polynomial iteration can help construct such vectors. After a Chebyshev iteration is applied to some initial vector v the resulting vector will have large components in any eigenvalue that is outside the best ellipse. If there is a small number of such eigenvalues in addition to the wanted ones $\lambda_1, \lambda_2, ..., \lambda_r$, then the Arnoldi projection process will compute them with a good accuracy and they will be used to correct the convex hull and the ellipse. Normally, in the next iteration, they should not appear again and others may possibly surge and will be added to the convex hull again. This can give an efficient adaptive and self correcting process. A few details of this combination which we refer to as the *enhanced initial vector approach* will be given in the next sections.

2.2. The Enhanced Initial Vector Approach

Suppose that we can find an ellipse $E(c, e, a)$ that contains all the eigenvalues of A except the r wanted ones, i.e., the r eigenvalues of A with largest real parts. We will describe in a moment an adaptive way of getting such an ellipse. Then an appealing algorithm would be to run a certain number of steps of the Chebyshev iteration and take the resulting vector z_k as initial vector in the Arnoldi process. From the Arnoldi purification process one obtains a set of m eigenvalues, r of which are approximations to the r wanted ones, as was suggested in the previous section, while the remaining ones will be useful for adaptively constructing the best ellipse. After a cycle consisting of k steps of the Chebyshev iteration followed by m steps of the purification process, the accuracy realized for the r rightmost eigenpairs may not be sufficient and restarting will then be necessary. The following is an outline of a simple algorithm based on the above ideas:

ALGORITHM **7.2 Arnoldi-Chebyshev**

1. **Start:** *Choose an initial vector* v_1, *a number of Arnoldi steps* m *and a number of Chebyshev steps* k.

2. **Iterate:**

   (a) *Perform* m *steps of the Arnoldi algorithm starting with* v_1. *Compute the* m *eigenvalues of the resulting Hessenberg matrix. Select the* r *eigenvalues of largest real parts* $\tilde{\lambda}_1, \cdots, \tilde{\lambda}_r$ *and take* $\tilde{R} = \{\tilde{\lambda}_{r+1}, \cdots, \tilde{\lambda}_m\}$. *If satisfied stop.*

   (b) *Using* $\tilde{R}$, *obtain the new estimates of the parameters* c *and* e *of the best ellipse. Then compute the initial vector* z_0 *for the Chebyshev iteration as a linear combination of the approximate eigenvectors* $\tilde{u}_i, i = 1, \cdots, r$.

   (c) *Perform* k *steps of the Chebyshev iteration to obtain* z_k. *Take* $v_1 = z_k/\|z_k\|$ *and go back to 1.*

Next, we will give some details on practicalities concerning the above algorithm.

2.3. Computing an Optimal Ellipse

As was explained earlier we would like to find the 'best' ellipse enclosing the set R of unwanted eigenvalues, i.e., the eigenvalues other than the ones with the r algebraically largest real parts. We must begin by clarifying what is meant by 'best' in the present context. Consider Figure 7.3 representing a spectrum of some matrix A and suppose that we are interested in the r rightmost eigenvalues, i.e., $r = 4$ in the figure.

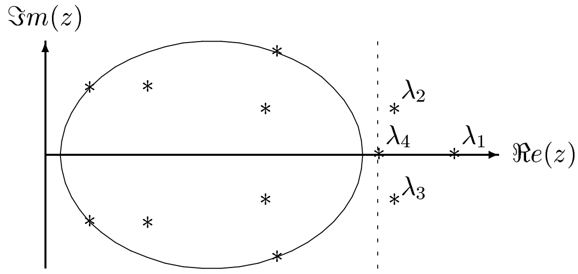

Figure 7.3 Example of a spectrum and the enclosing best ellipse for $r = 4$.

If $r = 1$ then one may simply seek the best ellipse in the sense of minimizing the convergence ratio $\tau(\lambda_1)$. This situation is identical to that of Chebyshev Iteration for linear systems for which much work has been done.

When $r > 1$, then we have several convergence ratios, each corresponding to one of the desired eigenvalues $\lambda_i, i = 1, \cdots, r$, and several possible strategies may be defined to try to optimize the process.

Initially, assume that λ_r is real (Figure 7.3) and consider any ellipse $E(c, e, a)$ including the set R of unwanted eigenvalues and not the eigenvalues

$$\{\lambda_1, \lambda_2, \cdots, \lambda_r\}.$$

It is easily seen from our comments of subsection 1.1 that if we draw a vertical line passing through the eigenvalue λ_r, all eigenvalues to the right of the line will converge faster than those to the left. Therefore, when λ_r is real, we may simply define the ellipse as the one that minimizes the convergence ratio of λ_r with respect to the two parameters c and e.

When λ_r is not real, the situation is more complicated. We could still attempt to maximize the convergence ratio for the eigenvalue λ_r, but the formulas giving the optimal ellipse do not readily extend to the case where λ_r is complex and the best ellipse becomes difficult to determine. But this is not the main reason why this choice is not suitable. A close look at Figure 7.3, in which we assume $r = 5$, reveals that the best ellipse for λ_r may not be a good ellipse for some of the desired eigenvalues. For example, in the figure the eigenvalues λ_2, λ_3 should be computed before the pair λ_4, λ_5 since their real parts are larger. However, because they are enclosed by the best ellipse for λ_5 they may not converge until many other eigenvalues will have converged including $\lambda_4, \lambda_5, \lambda_n, \lambda_{n-1}$ and possibly other unwanted eigenvalues not shown in the figure.

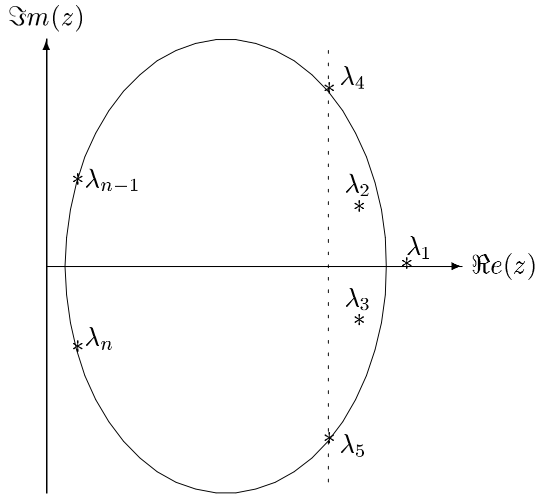

Figure 7.4 Case where $\mu = \lambda_r$ (complex): the eigenvalues λ_2 and λ_3 are inside the 'best' ellipse.

The difficulty comes from the fact that this strategy will not favor the eigenvalues with largest real parts but those belonging to the outermost confocal ellipse. It can be resolved by just maximizing the convergence ratio of λ_2 instead of λ_5 in this case. In a more complex situation it is unfortunately more difficult to determine at which particular eigenvalue λ_k or more generally at which value μ it is best to maximize $\tau(\mu)$. Clearly, one could solve the problem by taking $\mu = \Re e(\lambda_r)$, but this is likely to result in a suboptimal choice.

As an alternative, we can take advantage of the previous ellipse, i.e., an ellipse determined from previous purification steps. We determine a point μ on the real line *having the same convergence ratio as* λ_r*, with respect to the previous ellipse.* The next 'best' ellipse is then determined so as to maximize the convergence ratio for this point μ. This reduces to the previous choice $\mu = \Re e(\lambda_r)$ when λ_r is real. At the very first iteration one can set μ to be $\Re e(\lambda_r)$. This is illustrated in Figure 7.5. In Figure 7.5 the ellipse in solid is the optimal ellipse obtained from some previous calculation from the dynamic process. In dashed line is an ellipse that is confocal to the previous ellipse which passes through λ_r. The point μ is defined as one of the two points where this ellipse crosses the real axis.

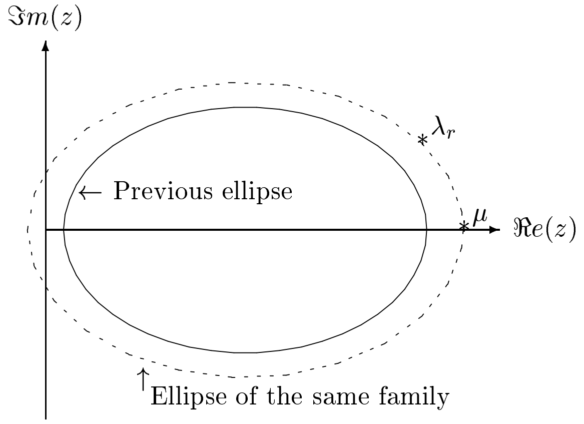

Figure 7.5 Point on the real axis whose convergence is equivalent with that of λ_r with respect to the previous ellipse.

The question which we have not yet fully answered concerns the practical determination of the best ellipse. At a typical step of the Arnoldi process we are given m approximations $\tilde{\lambda}_i, i = 1, \cdots, m$, of the eigenvalues of A. This approximate spectrum is divided in two parts: the r wanted eigenvalues $\tilde{\lambda}_1, \cdots, \tilde{\lambda}_r$ and the set $\tilde{R}$ of the remaining eigenvalues $\tilde{R} = \{\tilde{\lambda}_{r+1}, \cdots, \tilde{\lambda}_m\}$. From the previous ellipse and the previous sets $\tilde{R}$, we would like to determine the next estimates for the optimal parameters c and e.

A similar problem was solved in the context of linear systems of equations and the technique can easily be adapted to our situation. We refer the reader to the two articles by Manteuffel [99, 100]. The change of variables $\xi = (\mu - \lambda)$ easily transforms μ into the origin in the ξ–plane and the problem of maximizing the ratio $\tau(\mu)$ is transformed into one of maximizing a similar ratio in the ξ–plane for the origin, with respect to the parameters c and e. An effective technique for solving this problem has been developed in [98], [100] but its description is rather tedious and will be omitted. We only indicate that there exist reliable software that will deliver the optimal values of $\mu - c$ and e at output if given the shifted eigenvalues $\mu - \tilde{\lambda}_j, \quad j = r+1, \cdots, m$ on input.

We now wish to deal with a minor difficulty encountered when λ_1 is complex. Indeed, it was mentioned in Section 1 that the eigenvalue λ_1 in (7.6) should, in practice, be replaced by some approximation ν of λ_1. Initially, ν can be set to some initial guess. Then, as the approximation $\tilde{\lambda}_1$ computed from Algorithm 7.2 becomes available it can be used. If it is real then we can take $\nu = \tilde{\lambda}_1$ and the iteration can be carried out in real arithmetic as was already shown, even when e is purely imaginary. However, the iteration will become complex if $\tilde{\lambda}_1$ is complex. To avoid this it suffices to take ν to be one of the two points where the ellipse $E(c, e, a_1)$ passing through $\tilde{\lambda}_1$, crosses the real axis. The effect of the corresponding scaling of the Chebyshev polynomial will be identical with that using $\tilde{\lambda}_1$ but will present the advantage of avoiding complex arithmetic.

2.4. Starting the Chebyshev Iteration.

Once the optimal parameters c and e have been estimated we are ready to carry out a certain number k of steps of the Chebyshev iteration (7.9). In this subsection, we would like to indicate how to select the starting vector z_0 for this iteration. In the hybrid algorithm outlined in the previous section, the Chebyshev iteration comes after an Arnoldi step. It is then desirable to start the Chebyshev iteration by a vector which is a linear combination of the approximate eigenvectors associated with the rightmost r eigenvalues.

Let ξ_i be the coefficients of the desired linear combinations. Then the initial vector for the Chebyshev process is

$$z_0 = \sum_{i=1}^{r} \xi_i \tilde{u}_i = \sum_{i=1}^{r} \xi_i V_m \tilde{y}_i = V_m \left[\sum_{i=1}^{r} \xi_i \tilde{y}_i \right] . \tag{7.14}$$

Therefore, the eigenvectors $\tilde{u}_i, i = 1, \ldots, r$, need not be computed explicitly. We only need to compute the eigenvectors of the Hessenberg matrix H_m and to select the appropriate coefficients ξ_i. An important remark is that if we choose the ξ's to be real and such that $\xi_i = \xi_{i+1}$ for all conjugate pairs $\lambda_i, \lambda_{i+1} = \bar{\lambda}_i$, then the above vector z_0 is real.

Assume that all eigenvectors, exact and approximate, are normalized so that their 2–norms are equal to one. One desirable objective when choosing the above linear combination is to attempt to make z_k, the vector which starts the *next* Arnoldi step, equal to a sum of eigenvectors of A of norm unity, i.e., the objective is to have $z_k = \theta_1 u_1 + \theta_2 u_2 + \cdots + \theta_r u_r$, with $|\theta_i| = 1, i = 1, 2, \cdots, r$. For this purpose, suppose that for each approximate eigenvector $\tilde{u}_i = \gamma_i u_i + \epsilon_i$, where the vector ϵ_i has no components in $u_1, \cdots, u_r$. Then:

$$z_k = \xi_1 \gamma_1 u_1 + \xi_2 \gamma_2 u_2 + \cdots + \xi_r \gamma_r u_r + \epsilon,$$

where

$$\epsilon = \sum_{i=1}^{r} \xi_i \epsilon_i.$$

Near convergence $|\gamma_i|$ is close to one and $\|\epsilon_i\|$ is small. The result of k steps of the Chebyshev iteration applied to z_0 will be a vector z_k such that:

$$z_k \approx \xi_1 \gamma_1 u_1 + \kappa_2^k \xi_2 \gamma_2 u_2 + \cdots + \kappa_r^k \xi_r \gamma_r u_r + p_k(A)\epsilon. \tag{7.15}$$

Since ϵ has no components in $u_1, i = 1, \cdots, r$, $p_k(A)\epsilon$ tends to zero faster than the first r terms, as k tends to infinity. Hence, taking $\xi_i = \kappa_i^{-k}, i = 1, \cdots, r$, will give a vector which has components γ_i in the eigenvectors $u_i, i = 1, \cdots, r$. Since $|\gamma_i| \approx 1$ near convergence this is a satisfactory choice.

Another possibility is to weigh the combination of $\tilde{u}_i$ according to the accuracy obtained after an Arnoldi step, for example:

$$\xi_i = \|(A - \tilde{\lambda}_i I)\tilde{u}_i\|. \tag{7.16}$$

Notice that the residuals of two complex conjugate approximate eigenelements are equal, so this choice will also lead to a real z_0.

Finally, we would like to mention that an alternative is to compute one eigenvalue - Schur vector pair at a time and to proceed to an implicit deflation technique. From experience this alternative is far more reliable than one described in this section and avoids the difficulty of having to select the proper z_0 as a linear combination of the approximate eigenvectors. The deflated algorithm will be described shortly.

2.5. Choosing the Parameters m and k.

The number of Arnoldi steps m and the number of Chebyshev steps k are important parameters that affect the effectiveness of the method. Since we want to obtain more eigenvalues than the r desired ones, in order to use the remainder in choosing the parameters of the ellipse, m should be at least $r+2$ (to be able to compute a complex pair). In practice, however, it is preferable to take m several times larger than r. In typical runs m is at least $3r$ or $4r$ but can very well be even larger if storage is available. It is also possible to change m dynamically instead of keeping it fixed to a certain value but this variation will not be considered here.

When choosing k, we have to take into account a number of facts. First, taking k too small may slow down of the algorithm; ultimately when $k = 0$, the method becomes the simple iterative

Arnoldi method. On the other hand it may not be effective to pick k too large, otherwise the vector z_k may become nearly an eigenvector which could cause some numerical difficulties in the Arnoldi process. In addition, the parameters c, e, of the ellipse may be far from optimal and it is better to reevaluate them frequently.

Recalling that the component in the direction of u_1 will remain constant while those in $u_i, i = 2, \cdots, r$, will be of the same order as κ_i^k, we should attempt to avoid having a vector z_k which is entirely in the direction of u_1. This can be done by requiring that all $\kappa_i^k, i = 2, \cdots, r$, be no less than a certain tolerance δ, i.e.,

$$k \approx \ln(\delta) / \ln[\kappa_j], \tag{7.17}$$

where κ_j is the largest convergence ratio among $\kappa_i, i = 2, \cdots, r$. In our experimental codes we have opted to choose δ to be nearly the square root of the unit round–off.

Other practical factors should also enter into consideration. For example, it is desirable that a maximum number of Chebyshev steps $n_{\max}$ be fixed by the user. Also when we are close to convergence, we should avoid employing an unnecessarily large number of steps as might be dictated by a straightforward application of (7.17).

3. Deflated Arnoldi-Chebyshev

There are some disadvantages in the 'enhanced initial vector approach' discussed in the previous section. In particular, the process can be slow of even diverge in some cases when the eigenvalues are poorly separated. An alternative is to compute one eigenvalue-eigenvector pair at a time and proceed just as for the restarted Arnoldi method with deflation described in Chapter VI. The algorithm is in fact very similar in structure to Algorithm 6.4. The only difference is that the initial vector in the outer loop is now preprocessed by a Chebyshev acceleration.

The implementation uses a single basis $v_1, v_2, ..., v_m$ whose first vectors are the Schur vectors of A that have already converged. If the $\nu - 1$ vectors $v_1, v_2, ..., v_{\nu-1}$ have converged then we start by choosing a vector v_ν which is orthogonal to $v_1, ..., v_{\nu-1}$ and of norm 1. We then perform $m - \nu$ steps of an Arnoldi process, orthogonalizing the vector v_j against all previous $v_i's$, including $v_1, ..., v_{\nu-1}$. Finally, we restart as in the previous algorithm, taking v_1 to be $p_k(A)z_0$, where z_0 is the approximate Schur vector produced by the Arnoldi process. The algorithm is sketched below.

ALGORITHM **7.3 (Deflated Arnoldi-Chebyshev)**

A. Start: *Choose an initial vector* v_1 *of norm unity.*

B. Eigenvalue Loop: *For* $l = 1, 2, ..., p$ *do:*

1. *Arnoldi Iteration. For* $j = l, l + 1, ..., m$ *do:*
   - *Compute* $w := Av_j$*;*
   - *Compute a set of* j *coefficients* h_{ij} *such that* $w := w - \sum_{i=1}^{j} h_{ij} v_i$ *is orthogonal to all previous* v_i*'s,* $i = 1, 2, ..., j$*;*
   - *Compute* $h_{j+1,j} = \|w\|_2$ *and* $v_{j+1} = w/h_{j+1,j}$.
2. *Compute a desired Ritz pair* $\tilde{u}_l, \tilde{u}_l$*, and corresponding residual norm* ρ_l.
3. *Update the convex hull of* R*. Obtain new estimates for* c *and* e*. Compute next candidate eigenvalue and corresponding eigenvector* $\tilde{u}$*. Define* $z_0 = \tilde{u}$.
4. *Compute* $z_k = p_k(A)z_0$.
5. *Orthonormalize* z_k *against all previous* v_j*'s to get the approximate Schur vector* $\tilde{u}_l$ *and define* $v_l := \tilde{u}_l$.
6. *If* ρ_j *is small enough then accept* $\tilde{v}_l$ *as the next Schur vector, compute* $h_{i,l} = (Av_l, v_i)$ $i = 1, .., l$*. Else go to (B.1).*

Recall that in the B-loop, the Schur vectors associated with the eigenvalues $\lambda_1, ..., \lambda_{l-1}$ are frozen and so is the corresponding upper triangular matrix corresponding to these vectors.

4. Chebyshev Subspace Iteration

We will use the same notation as in the previous sections. Suppose that we are interested in the rightmost r eigenvalues and that the ellipse $E(c, e, a)$ contains the set R of all the remaining eigenvalues. Then the principle of the Chebyshev acceleration of subspace iteration is simply to replace the powers A^k in the first part of the basic algorithm 5.1 described in Chapter V, by $p_k(A)$ where p_k is the polynomial defined by (7.5). It can be shown that the approximate eigenvector $\tilde{u}_i, i = 1, \cdots, r$ converges towards u_i, as $C_k(a/e)/C_k[(\lambda_i - c)/e]$, which, using arguments similar to those of subsection (1.1) is equivalent to η_i^k where

$$\eta_i = \frac{a + [a^2 - 1]^{1/2}}{a_i + [a_i^2 - 1]^{1/2}}. \tag{7.18}$$

The above convergence ratio can be far superior to the standard ratio $|\lambda_{r+1}/\lambda_i|$ which is achieved by the non-accelerated algorithm. However, we recall that subspace iteration computes the eigenvalues of largest moduli. Therefore, the unaccelerated and the accelerated subspace iteration methods are not always comparable since they achieve different objectives.

On the practical side, the best ellipse is obtained dynamically in the same way as was proposed for the Chebyshev–Arnoldi process. The accelerated algorithm will then have the following form.

ALGORITHM **7.4** *Chebyshev Subspace Iteration*

1. **Start:** $Q \leftarrow X$.

2. **Iterate:** *Compute* $Q \leftarrow p_k(A)Q$.

3. **Project:** *Orthonormalize Q and get eigenvalues and Schur vectors of $C = Q^T AQ$. Compute $Q \leftarrow QF$, where F is the matrix of Schur vectors of C.*

4. **Test for convergence:** *If Q is a satisfactory set of Schur vectors then stop, else get new best ellipse and go to 2.*

Most of the ideas described for the Arnoldi process extend naturally to this algorithm, and we now discuss briefly a few of them.

4.1. Getting the Best Ellipse.

The construction of the best ellipse is identical with that seen in subsection 2.3. The only difficulty we might encounter is that the additional eigenvalues which are used to build the best ellipse may now be far less accurate than those provided by the more powerful Arnoldi technique. More care must therefore be taken in order to avoid building an ellipse based on too inaccurate eigenvalues as this may cause substantial slow down in convergence.

4.2. Parameters k and m.

Here, one can take advantage of the abundant work on subspace iteration available in the literature. All we have to do is replace the convergences $|\lambda_{r+1}/\lambda_i|$ of the basic subspace iteration by the new ratios η_i of (7.18). For example, one way to determine the number of Chebyshev steps k, proposed in Rutishauser [137] and in Jennings and Stewart [77] is

$$n \approx \frac{1}{2}[1 + \ln(\epsilon^{-1}) / \ln(\eta_1)],$$

where ϵ is some parameter depending on the unit round–off. The goal of this choice is to prevent the rounding errors from growing beyond the level of the error in the most slowly converging eigenvector. The parameter k is also limited from above by a user supplied bound $n_{\max}$, and by the fact that if we are close to convergence a smaller k can be determined to ensure convergence at the next projection step.

The same comments as in the Arnoldi–Chebyshev method can be made concerning the choice of m, namely that m should be at least $r+2$, but preferably even larger although in a lesser extent than for Arnoldi. For the symmetric case it is often suggested to take to be a small multiple of r, e.g., $m = 2r$ or $m = 3r$.

4.3. Deflation

Another special feature of the subspace iteration is the deflation technique which consists of working only with the nonconverged eigenvectors, thus 'locking' those that have already converged. Clearly, this can be used in the accelerated subspace iteration as well and will enhance its efficiency. For the more stable versions such as those based on Schur vectors, a similar device can be applied to the Schur vectors instead of the eigenvectors.

5. Least Squares - Arnoldi

The choice of ellipses as enclosing regions in Chebyshev acceleration may be overly restrictive and ineffective if the shape of the convex hull of the unwanted eigenvalues bears little resemblance with an ellipse. This has spurred much research in which the acceleration polynomial is chosen so as to minimize an L_2-norm of the polynomial p on the boundary of the convex hull of the unwanted eigenvalues with respect to some suitable weight function ω. The only restriction with this technique is that the degree of the polynomial is limited because of cost and storage requirement. This, however, is overcome by compounding low degree polynomials. The stability of the computation is enhanced by employing a Chebyshev basis and by a careful implementation in which the degree of the polynomial is taken to be the largest one for which the Gram matrix has a tolerable conditioning. The method for computing the least squares polynomial is fully described in [142] but we present a summary of its main features below.

5.1. The Least Squares Polynomial

Suppose that we are interested in computing the r eigenvalues of largest real parts $\lambda_1, \lambda_2, \ldots \lambda_r$ and consider the vector

$$z_k = p_k(A) z_0 \tag{7.19}$$

where p_k is a degree k polynomial. Referring to the expansion (7.2) we wish to choose among all polynomials p of degree $\le k$ one for which $p(\lambda_i), i{>}r$ are small relative to $p(\lambda_i), i \le r$. Assume that by some adaptive process, a polygonal region H which encloses the remaining eigenvalues becomes available to us. We then arrive at the problem of approximation theory which consists of finding a polynomial of degree k whose value inside some (polygonal) region is small while its values at r particular points (possibly complex) outside the region are large. For simplicity we start with the case where $r = 1$, i.e., only the eigenvalue λ_1 and its associated eigenvectors are sought. We seek a polynomial that is large at λ_1 and small elsewhere. For convenience we can always normalize the polynomial so that

$$p_k(\lambda_1) = 1. \tag{7.20}$$

The desired polynomial satisfying the above constraint can be sought in the form

$$p_k(\lambda) \equiv 1 - (\lambda - \lambda_1) s_k(\lambda) \tag{7.21}$$

where s_k is a polynomial of degree $k - 1$.

Since it is known that the maximum modulus of an analytic function over a region of the complex plane is reached on the boundary of the region, one solution to the above problem is to minimize an L_2-norm associated with some weight function ω, over all polynomials of degree k satisfying the constraint (7.20). We need to choose a weight function ω that will facilitate practical computations.

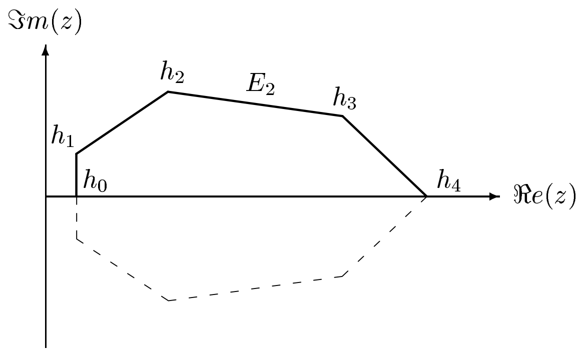

Figure 7.6 Polygon H containing the spectrum of A.

Let the region H of the complex plane, containing the eigenvalues $\lambda_{r+1}, \ldots \lambda_n$, be a polygon consisting of μ edges $E_1, E_2, \ldots E_\mu$, each edge E_j linking two successive vertices h_{j-1} and h_j of H, see Figure 7.6. Denoting by $c_j = \frac{1}{2}(h_j + h_{j-1})$ the center of the edge E_j and by $d_j = \frac{1}{2}(h_j - h_{j-1})$ its half-width, we define the following Chebyshev weight function on each edge:

$$\omega_j(\lambda) = \frac{2}{\pi}|d_j^2 - (\lambda - c_j)^2|^{-1/2}. \tag{7.22}$$

The weight ω on the boundary ∂H of the polygonal region is defined as the function whose restriction to each edge E_j is ω_j. Finally, the L_2-inner-product over ∂H is defined by

$$\begin{aligned} < p, q >_\omega &= \int_{\partial H} p(\lambda)\overline{q(\lambda)}\omega(\lambda)|d\lambda| & (7.23) \\ &= \sum_{j=1}^{\mu} \int_{E_j} p(\lambda)\overline{q(\lambda)}\omega_j(\lambda)|d\lambda|, & (7.24) \end{aligned}$$

and the corresponding L_2-norm is denoted by $\|.\|_\omega$.

Often, the matrix A is real and the convex hull may be taken to be symmetric with respect to the real axis. In this situation it is better to define the convex hull as the union of two polygons H^+ and H^- which are symmetric to each other. These two are represented in solid line and dashed line respectively in the figure 7.6. Then, when the coefficients of p and q are *real*, we only need to compute the integrals over the edges of the upper part H^+ of H because of the relation

$$< p, q >_\omega = 2\Re e \left[\int_{\partial H^+} p(\lambda)\overline{q(\lambda)}\omega(\lambda)|d\lambda| \right] . \tag{7.25}$$

Having defined an inner product we now define in the simplest case where $r = 1$, the 'least-squares' polynomial that minimizes

$$\|1 - (\lambda - \lambda_1)s(\lambda)\|_\omega. \tag{7.26}$$

Note that there are other ways of defining the least squares polynomial. Assume that we use a certain basis $t_0, \ldots, t_{k-1}$. and let us express the degree $k-1$ polynomial $s(\lambda)$ in this basis as

$$s(\lambda) = \sum_{j=0}^{k-1} \eta_j t_j(\lambda) . \tag{7.27}$$

Each polynomial $(\lambda - \lambda_1)t_j(\lambda)$ is of degree $j+1$ and can be expressed as

$$(\lambda - \lambda_1)t_j(\lambda) = \sum_{i=0}^{j+1} \tau_{ij} t_i(\lambda)$$

Denote by η the vector of the η_j's for $j = 0, \ldots, k-1$ and by γ the vector of coefficients $\gamma_j, j = 0, \ldots, k$ of $(\lambda - \lambda_1)s(\lambda)$ in the basis $t_0, \ldots, t_k$ and define $\tau_{ij} = 0$ for $i > j+1$. Then the above two relations state that

$$(\lambda - \lambda_1)s(\lambda) = \sum_{j=0}^{k-1} \eta_j \sum_{i=0}^{k} \tau_{ij} t_i(\lambda) = \sum_{i=0}^{k} \left(\sum_{j=0}^{k-1} \tau_{ij}\eta_j \right) t_i(\lambda)$$

In matrix form this means that

$$\gamma = T_k \eta$$

where T_k is the $(k+1) \times k$ matrix of coefficients t_{ij}'s, which is upper Hessenberg. In fact, it will seen that the matrix T_k is tridiagonal when Chebyshev bases are used.

The least-squares problem (7.26) will translate into a linear least-squares problem for the vector η. We will discuss some of the details of this approach next. There are two critical parts in this technique. The first concerns the choice of the basis and the second concerns the solution least-squares problem.

5.2. Use of Chebyshev Bases

To motivate our choice of the basis $\{t_j\}$, we assume at first that the best polynomial is expressed in the 'power' basis

$$1, \lambda, \lambda^2, \cdots \ .$$

Then, the solution of the least-squares problem (7.26) requires the factorization of the Gram matrix consisting of all the inner products $< \lambda^{i-1}, \lambda^{j-1} >_\omega$:

$$M_k = \{< t_j, t_i >_\omega\}_{i,j=0,\ldots,k}.$$

This matrix, often referred to as the moment matrix , can become extremely ill-conditioned and methods based on the use of the power basis will generally be limited to low degree calculations, typically not exceeding 10. A more reliable alternative is to replace the basis $\{\lambda^{i-1}\}$ by a more stable basis. One such basis, well understood in the real case, is that consisting of Chebyshev polynomials over an ellipse that contains the convex hull. The solution polynomial (7.40) will be expressed in terms of a Chebyshev basis associated with the ellipse of smallest area containing H. Such an ellipse is illustrated in Figure 7.7.

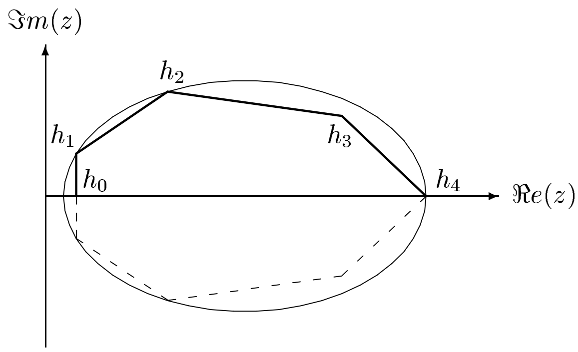

Figure 7.7 The ellipse of smallest area containing the convex hull of $\sigma(A)$.

Computing the ellipse of smallest area that encloses H is a rather easy task, far easier than that of computing ellipses which minimize convergence rates for Chebyshev acceleration, see Exercise P-7.3 for details.

5.3. The Gram Matrix

The next step in the procedure for computing the best polynomial, is to evaluate the Gram matrix M_k. For the Chebyshev basis, the Gram matrix M_k can be constructed recursively *without any numerical integration.*

The entries of the Gram matrix are defined by,

$$m_{ij} = < t_{j-1}, t_{i-1} >_\omega \ , \quad i, j = 1, \ldots, k+1 \ .$$

Note that because of the symmetry of the domain, the matrix M_k has real coefficients. We start by expressing each polynomial $t_i(\lambda)$ in terms of the Chebyshev polynomials

$$C_l\left(\frac{\lambda - c_\nu}{d_\nu}\right) \equiv C_l(\xi)$$

for each of the μ edges E_ν, $\nu = 1, \ldots, \mu$. The variable ξ takes real values when λ lies on the edge E_ν. In other words we express each t_i as

$$t_i(\lambda) = \sum_{l=0}^{i} \gamma_{l,\nu}^{(i)} C_l(\xi) \ , \tag{7.28}$$

$$\xi = \frac{\lambda - c_\nu}{d_\nu} \ . \tag{7.29}$$

Each polynomial t_i will have μ different expressions of this type, one for each edge E_ν. Clearly, these expressions are redundant since one of them is normally enough to fully determine the polynomial t_i. However, this redundancy is useful from the practical point of view as it allows to perform an efficient computation in a stable manner. The following proposition enables us to compute the Gram matrix from the expressions (7.28).

Proposition 7.1 *Assuming the expressions* (7.28) *for each of the polynomials t_i, the coefficients of the Gram matrix M_k are given by*

$$m_{i+1,j+1} = 2\ \Re e \left\{ \sum_{\nu=1}^{\mu} \left(2\ \gamma_{0,\nu}^{(i)} \overline{\gamma_{0,\nu}^{(j)}} + \sum_{l=1}^{j} \gamma_{l,\nu}^{(i)} \overline{\gamma_{l,\nu}^{(j)}} \right) \right\} \ , \tag{7.30}$$

for all i, j such that $0 \le i \le j \le k$.

Proof. The result is a simple consequence of the orthogonality of the Chebyshev polynomials, the change of variables (7.29) and the expression (7.25). ∎

We now need to be able to compute the expansion coefficients. Because of the three term-recurrence of the Chebyshev polynomials it is possible to carry the computation of these coefficients in a recursive manner. We rewrite the recurrence relation for the shifted Chebyshev polynomials in the form

$$\beta_{i+1} t_{i+1}(\lambda) = (\lambda - \alpha_i) t_i(\lambda) - \delta_i t_{i-1}(\lambda), i = 0, 1, \ldots, k, \ldots, \tag{7.31}$$

with the convention that $t_{-1} \equiv= 0$ and $\delta_0 = 0$. Using the definitions (7.28) and (7.29), we get for each edge,

$$\beta_{i+1}t_{i+1}(\lambda) = (d_\nu \xi + c_\nu - \alpha_i)\sum_{l=0}^{i} \gamma_{l,\nu}^{(i)} C_l(\xi) - \delta_i \sum_{l=0}^{i-1} \gamma_{l,\nu}^{(i-1)} C_l(\xi)$$

which provides the expressions for t_{i+1} from those of t_i and t_{i-1} by exploiting the relations

$$\begin{aligned} \xi C_l(\xi) &= \frac{1}{2}[C_{l+1}(\xi) + C_{l-1}(\xi)] \quad l > 0 \ , \\ \xi C_0(\xi) &= C_1(\xi) \ . \end{aligned}$$

The result is expressed in the following proposition.

Proposition 7.2 *For $\nu = 1, 2, \ldots \mu$, the expansion coefficients $\gamma_{l,\nu}^{(i)}$ satisfy the recurrence relation,*

$$\beta_{i+1}\gamma_{l,\nu}^{(i+1)} = \frac{d_\nu}{2}\left[\gamma_{l+1,\nu}^{(i)} + \gamma_{l-1,\nu}^{(i)}\right] + (c_\nu - \alpha_i)\gamma_{l,\nu}^{(i)} - \delta_i \gamma_{l,\nu}^{(i-1)} \quad (7.32)$$

for $l = 0, 1, \ldots, i+1$ with the notational convention,

$$\gamma_{-1,\nu}^{(i)} \equiv \gamma_{1,\nu}^{(i)}, \quad \gamma_{l,\nu}^{(i)} = 0 \quad \textit{for} \quad l>i \ .$$

The total number of operations required for computing a Gram matrix with the help of the above two propositions is $O(\mu k^3/3)$. This cost may be high for high degree polynomials. However, this cost will in general not be significant relatively to the total number of operations required with the matrix A which is typically very large. It is also not recommended to compute least squares polynomials of degree higher than 40 or 50.

5.4. Computing the Best Polynomial

In the simple case where we are attempting to compute the eigenvalue λ_1 and the associated eigenvector, we need to compute the polynomial $s(\lambda)$ that minimizes the norm

$$J(\eta) = \|1 - (\lambda - \lambda_1)s(\lambda)\|_w \quad (7.33)$$

where $s(\lambda)$ is the unknown polynomial of degree $k-1$ expressed in the form (7.27).

Let T_k be the $(k+1)\times k$ tridiagonal matrix

$$T_k = \begin{pmatrix} \alpha_0 & \delta_1 & & & \\ \beta_1 & \alpha_1 & \delta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-2} & \delta_{k-1} \\ & & & \beta_{k-1} & \alpha_{k-1} \\ & & & & \beta_k \end{pmatrix} \tag{7.34}$$

whose coefficients $\alpha_i, \delta_i, \beta_i$ are those of the three-term recurrence (7.31). Given two polynomials of degree k

$$p(\lambda) = \sum_{i=0}^{k} \gamma_i t_i(\lambda) \quad \text{and} \quad q(\lambda) = \sum_{i=0}^{k} \theta_i t_i(\lambda)$$

it is easy to show that the inner product of these two polynomials can be computed from

$$< p, q >_\omega = (M_k \gamma, \theta) \tag{7.35}$$

where $\gamma = (\gamma_0, \gamma_1, \ldots, \gamma_k)^T$ and $\theta = (\theta_0, \theta_1, \ldots, \theta_k)^T$. Therefore, an alternative expression for $J(\eta)$ is

$$J(\eta)^2 = [e_1 - (T_k - \lambda_1 I)\eta]^H M_k [e_1 - (T_k - \lambda_1 I)\eta]$$

and as a consequence, we can prove the following theorem.

Theorem 7.1 *Let*

$$M_k = LL^T$$

be the Choleski factorization of the $(k+1)\times(k+1)$ *Gram matrix* M_k *and denote by* H_k *the* $(k+1)\times k$ *upper Hessenberg matrix*

$$H_k = L^T (T_k - \lambda_1 I),$$

where T_k *is the tridiagonal matrix (7.34) defined from the three-term recurrence of the basis* t_i*. Then the function* $J(\eta)$ *satisfies the relation,*

$$J(\eta) = \|l_{11} e_1 - H_k \eta\|_2. \tag{7.36}$$

Therefore the computation of the best polynomial requires the solution of a $(k+1) \times k$ least squares problem. This is best done by reducing the Hessenberg matrix H_k into upper triangular form by using Givens rotations.

The above theorem does not deal with the case where we have several eigenvalues to compute, i.e., with the case $r > 1$. For this situation, we need to redefine the problem slightly. The following development is also of interest because it gives an alternative formulation to the least squares polynomial even for the case $r = 1$.

We start by introducing what is referred to as *kernel polynomials*,

$$K_k(\xi, \lambda) = \sum_{j=0}^{k} \overline{\pi_j(\xi)} \pi_j(\lambda) \tag{7.37}$$

in which the π_j's are the orthogonal polynomials with respect to the appropriate inner product, here $< .,. >_\omega$. Then the following well-known theorem holds [31].

Theorem 7.2 *Among all polynomials of degree k normalized so that $p(\lambda_1) = 1$, the one with the smallest ω-norm is given by*

$$q_k(\lambda) = \frac{K_k(\lambda_1, \lambda)}{K_k(\lambda_1, \lambda_1)}. \tag{7.38}$$

This gives an interesting alternative to the polynomial derived previously. We will now generalize this result and discuss its practical implementation.

We begin by generalizing the constraint (7.20) by normalizing the polynomial at the points $\lambda_1, \lambda_2, \ldots, \lambda_r$ as follows,

$$\sum_{j=1}^{r} \mu_j p(\lambda_j) = 1 \tag{7.39}$$

in which the μ_j's , $j = 1, \ldots r$ constitute r different weights.

Then we have the following generalization of the above theorem.

Theorem 7.3 *Let $\{\pi_i\}_{i=0,\ldots,k}$ be the first $k+1$ orthonormal polynomials with respect to the L_2-inner-product (7.24). Then among all polynomials p of degree k satisfying the constraint (7.39), the one with smallest ω-norm is given by*

$$p_k(\lambda) = \frac{\sum_{i=0}^{k} \phi_i \pi_i(\lambda)}{\sum_{i=0}^{k} |\phi_i|^2} \ , \tag{7.40}$$

where $\phi_i = \overline{\sum_{j=1}^{r} \mu_j \pi_i(\lambda_j)}$.

Proof. We recall the reproducing property of kernel polynomials [31],

$$< p, K_k(\xi, \lambda) >_\omega = p(\xi) \ , \tag{7.41}$$

in which the integration is with respect to the variable λ. It is easily verified that the polynomial (7.40) satisfies the constraint (7.39) and that p_k can be recast as

$$p_k(\lambda) = C \sum_{j=0}^{k} \overline{\mu_j} K_k(\lambda_j, \lambda) \tag{7.42}$$

where C is some constant. Next, we consider any polynomial p satisfying the constraint (7.39) and write p in the form

$$p(\lambda) \ = \ p_k(\lambda) + E(\lambda),$$

from which we get,

$$\|p\|_\omega^2 = \|p_k\|_\omega^2 + \|E\|_\omega^2 + 2\Re e\{< E, p_k >_\omega\}. \tag{7.43}$$

Since both p and p_k satisfy the constraint (7.39) we must have

$$\sum_{j=1}^{r} \mu_j E(\lambda_j) = 0. \tag{7.44}$$

From (7.42) and from the reproducing property (7.41) we see that

$$\begin{aligned} < E, p_k >_\omega \ &= \ C \sum_{j=1}^{r} \mu_j < E, K_k(\lambda_j, \lambda) >_\omega \\ &= \ C \sum_{j=1}^{r} \mu_j E(\lambda_j) \ . \end{aligned}$$

Hence, from (7.44) $< E, p_k >_\omega = 0$ and (7.43) shows that $\|p\|_\omega \geq \|p_k\|_\omega$ for any p of degree $\leq k$. ■

As is now explained, the practical computation of the best polynomial p_k can be carried out by solving a linear system with the Gram matrix M_k. We could also compute the orthogonal polynomials π_j and take their linear combination (7.40) but this would not be as economical.

We consider the unscaled version of the polynomial (7.40) used in (7.42),

$$\hat{p}_k(\lambda) = \sum_{j=1}^{r} \bar{\mu}_j K_k(\lambda_j, \lambda) \ , \tag{7.45}$$

which satisfies a property stated in the next proposition.

Proposition 7.3 *Let t be the $(k+1)$-vector with components*

$$\tau_i = \sum_{j=1}^{r} \mu_j t_{i-1}(\lambda_j) \ , \quad i = 0, \ldots, k \ .$$

Then the coefficients of the polynomial $\hat{p}_k$ in the basis $\{t_j\}$ are the conjugates of the components of the k-vector,

$$\eta = M_k^{-1} t \ .$$

Proof. Consider the Choleski factorization $M_k = LL^T$ of the Gram matrix M_k. If we represent by $\underline{p}(\lambda)$ and $\underline{t}(\lambda)$ the vectors of size $k+1$ defined by

$$\underline{p}(\lambda) = (\pi_0(\lambda), \pi_1(\lambda), \ldots, \pi_k(\lambda))^T$$

and

$$\underline{t}(\lambda) = (t_0(\lambda), t_1(\lambda), \ldots, t_k(\lambda))^T$$

then we have the important relation,

$$\underline{p}(\lambda) = L^{-1} \underline{t}(\lambda) \tag{7.46}$$

which can be easily verified from (7.35). Notice that $K_k(\xi,\eta) = (\underline{p}(\lambda), \underline{p}(\xi))$ where $(.,.)$ is the complex inner product in $\mathbb{C}^{k+1}$, and therefore, from (7.45) and (7.46) we get

$$
\begin{aligned}
\hat{p}_k(\lambda) &= \sum_{j=1}^{r} \bar{\mu}_j \left(\underline{p}(\lambda), \underline{p}(\lambda_j)\right) \\
&= \sum_{j=1}^{r} \bar{\mu}_j \left(L^{-1}\underline{t}(\lambda), L^{-1}\underline{t}(\lambda_j)\right) = \sum_{j=1}^{r} \bar{\mu}_j \left(\underline{t}(\lambda), M_k^{-1}\underline{t}(\lambda_j)\right) \\
&= \left(\underline{t}(\lambda), M_k^{-1} \sum_{j=1}^{r} \mu_j \underline{t}(\lambda_j)\right) = \left(\underline{t}(\lambda), M_k^{-1} t\right) \\
&= \sum_{l=1}^{k+1} \bar{\eta}_l t_{l-1}(\lambda) \ ,
\end{aligned}
$$

which completes the proof. ∎

The proposition allows to avoid computing the orthogonal polynomials and to obtain the best polynomial directly in the desired basis. Finally, we point out that since the matrix M_k is real, if the τ_i's are real then the coefficient vector η is real if the λ_j's are selected in pairs of conjugate complex numbers.

5.5. Least Squares Arnoldi Algorithms

A resulting hybrid method similar to the Chebyshev Arnoldi Algorithm can be easily derived. The algorithm for computing the r eigenvalues with largest real parts is outlined next.

ALGORITHM **7.5 Least Squares Arnoldi Algorithm**

1. **Start:** *Choose the degree k of the polynomial p_k, the dimension m of the Arnoldi subspaces and an initial vector v_1.*

2. **Projection step:**

(a) *Using the initial vector* v_1*, perform* m *steps of the Arnoldi method and get the* m *approximate eigenvalues* $\{\tilde{\lambda}_1, \ldots \tilde{\lambda}_m\}$ *of the matrix* H_m*.*

(b) *Estimate the residual norms* $\rho_i, i = 1, \ldots, r$*, associated with the* r *eigenvalues of largest real parts* $\{\tilde{\lambda}_1, \ldots \tilde{\lambda}_r\}$ *If satisfied then Stop.*

(c) *Adapt: From the previous convex hull and the set* $\{\tilde{\lambda}_{r+1}, \ldots \tilde{\lambda}_m\}$ *construct a new convex hull of the unwanted eigenvalues.*

(d) *Obtain the new least squares polynomial of degree* k*.*

(e) *Compute a linear combination* z_0 *of the approximate eigenvectors* $\tilde{u}_i, i = 1, \ldots, r$*.*

3. **Polynomial iteration:**

   Compute $z_k = p_k(A)z_0$*. Compute* $v_1 = z_k/\|z_k\|$ *and goto 2.*

As can be seen the only difference with the Chebyshev algorithm is that the polynomial must now be computed. We must explain how the vector z_k can be computed. We will call w_i the auxiliary sequence of vectors $w_i = t_i(A)z_0$. One possibility would be to compute all the w_i's first and then accumulate their linear combination to obtain z_k. However, the w_i can also be accumulated at the same time as they are computed. More precisely, we can use a coupled recurrence as described in the next algorithm.

ALGORITHM **7.6 (For Computing** $z_k = p_k(A)z_0$**)**

1. **Start:** $\delta_0 := 0$*,* w_0*,* $y_0 = \eta_0 z_0$*.*

2. **Iterate:** *For* $i = 1, 2, \ldots, k$ *do:*

$$\begin{aligned} w_{i+1} &= \frac{1}{\beta_{i+1}}\left[(A - \alpha_i I)w_i - \delta_i w_{i-1}\right] \ , \\ y_i &= y_{i-1} + \eta_i w_{i+1} \ . \end{aligned}$$

3. **Finish:** $z_k = y_k$*.*

The intermediate vectors y_i are not related to the vectors z_i only the last vector y_k is.

We cannot, for reasons of space, describe all the details of the implementation. However, we mention that the linear combination at the end of step 3, is usually taken as follows:

$$z_0 = \sum_{i=1}^{r} \rho_i \tilde{u}_i$$

as for the Chebyshev iteration. Note that it is difficult, in general, to choose a linear combination that leads to balanced convergence because it is hard to represent a whole subspace by a single vector. This translates into divergence in many cases especially when the number of wanted eigenvalues r is not small. There is always the possibility of increasing the space dimension m, at a high cost, to ensure convergence but this solution is not always satisfactory from the practical point of view. Use of deflation constitutes a good remedy against this difficulty because it allows to compute one eigenvalue at a time which is much easier than computing a few of them at once. We omit the description of the corresponding algorithm whose general structure is identical with that of Algorithm 7.3.

One attractive feature of the deflation techniques is that the information gathered from the determination of the eigenvalue λ_i is still useful when iterating to compute the eigenvalue λ_{i+1}. The simplest way in which the information can be exploited is by using at least part of the convex hull determined during the computation of λ_i. Moreover, a rough approximate eigenvector associated with λ_{i+1} can be inexpensively determined during the computation of the eigenvalue λ_i and then used as initial vector in the next step for computing λ_{i+1}.

Another solution is to improve the separation of the desired eigenvalues by replacing A by a polynomial in A. This will be seen in the next chapter.

PROBLEMS

P-7.1 Prove that the relation (7.25) holds when the polynomials p and q are real and the polygon is symmetric with respect to the real line.

P-7.2 Show that the recurrence (7.8)-(7.9) can be performed in real arithmetic when A is real but e is complex. Rewrite the recurrence accordingly.

P-7.3 The purpose of this exercise is to develop formulas for the ellipse $E(c, e, a)$ of smallest area enclosing a polygon H. It is assumed that the polygon is symmetric about the real axis. Therefore the ellipse is also symmetric about the real axis. The following result will be assumed, see for example [99]: The best ellipse is either an ellipse that passes through 3 vertices of H and encloses H or an ellipse of smallest area passing through two vertices of H. Formulas for the first case have been established in the literature, see Manteuffel [99]. Therefore, we must only consider the second case. Let $\lambda_1 = (x_1, y_1)$ and $\lambda_2 = (x_2, y_2)$ two points in $\mathbb{R}^2$. We set

$$A = \frac{1}{2}(x2 - x1), \quad B = \frac{1}{2}(x1 + x2),$$
$$S = \frac{1}{2}(y2 - y1), \quad T = \frac{1}{2}(y1 + y2)$$

and define the variable $z = c - B$. At first, assume that $S \neq 0$ and define $Q = (S/T + T/S)/2$. Show that for a given z (which defines c) the only ellipse that passes through λ_1, λ_2 is defined by

$$\begin{aligned} e^2 &= \frac{1}{z}\left[(z + AT/S)(z + AS/T)(z - ST/A)\right] \\ a^2 &= (z + AT/S)(z + AS/T) \ . \end{aligned}$$

Then show that the optimal z is given by

$$z = \frac{A}{\sqrt{Q^2 + 3} \pm Q}$$

where $\pm$ is the sign of AS. In the particular case where $S = 0$ the above formulas break down. But then $c = B$ and one is lead to minimize the area as a function of a. Show that the minimum is reached for $a^2 = 2A^2$ and that the corresponding d is given by $d^2 = 2(A^2 - T^2)$.

P-7.4 Polynomials of degree 2 can be used to calculate intermediate eigenvalues of Hermitian matrices. Suppose we label the eigenvalues increasingly and that we have estimates for $\lambda_1, \lambda_{i-1}, \lambda_i, \lambda_{i+1}, \lambda_n$. Consider the family of quadratic polynomials that take the value 1 at λ_i and whose derivative at λ_i is zero. Find one such polynomial that will be suitable for computing λ_i and the associated eigenvector. Is this a good polynomial.
Find a good polynomial for computing the eigenvalue λ_i.

P-7.5 Establish formula (7.35).

P-7.6 Prove Theorem 7.1.

NOTES AND REFERENCES. The contents in this Chapter are taken mostly from Saad [143, 142, 144, 147, 141]. The idea of Chebyshev acceleration for eigenvalue problems is an old one and seems to have been first advocated by Flanders and Shortley [47]. However, in a work that has been vastly ignored, Lanczos also did some very interesting contemporary work in acceleration technique [90], see also the related paper [93]. Lanczos' approach is radically different from that of Flanders and Shortley, which is the approach most numerical analysts are familiar with. Concerned about the difficulty in getting eigenvalue estimates, Lanczos proposed as an alternative to compute a polynomial approximation to the Dirac function based on the wanted eigenvalue. The approximation is made over an interval containing the spectrum, which can easily be obtained from Gerschgorin estimates. This turns out to lead to the so-called Fejer kernel in the theory of approximation by trigonometric functions and then naturally to Chebyshev polynomials. His approach is a least squares technique akin to the one proposed by Stiefel [173] and later Saad [142]. Some ideas on implementing Chebyshev acceleration in the complex plane were introduced by Wrigley [186] but the technique did not mature until the 1975 PhD thesis by Manteuffel [98] in which a FORTRAN implementation for solving linear systems appeared. The work in [143] was based on adapting Manteuffel's implementation for the eigenvalue problem. The least squares polynomial approach presented in this chapter is based on the technical report [142] and its revised published version [144]. In my experience, the least squares approach does seem to perform slightly better in practice than the Chebyshev approach. Its drawbacks (mainly, having to use relatively low degree polynomials) are rather minor in practice. ♠

Chapter VIII

Preconditioning Techniques

The notion of preconditioning is better known for linear systems than it is for eigenvalue problems. A typical preconditioned iterative method for linear systems amounts to replacing the original linear system $Ax = b$ by (for example) the equivalent system $B^{-1}Ax = B^{-1}b$, where B is a matrix close to A in some sense and defined as the product of a lower by an upper sparse triangular matrices. This equivalent system is then handled by a Krylov subspace method. For eigenvalue problems, the best known preconditioning is the so-called shift-and-invert technique which we already mentioned in Chapter IV. If the shift σ is suitably chosen the shifted and inverted matrix $B = (A - \sigma I)^{-1}$, will have a spectrum with much better separation properties than that of the original matrix A and this will result in faster convergence. The term 'preconditioning' here is quite appropriate since the better separation of the eigenvalues around the desired eigenvalue implies that the corresponding eigenvector is likely to be better conditioned.

1. Shift-and-invert Preconditioning

One of the most effective techniques for solving large eigenvalue problems is to iterate with the shifted and inverted matrix,

$$(A - \sigma I)^{-1} \tag{8.1}$$

for standard problems and with (for example)

$$(A - \sigma B)^{-1} B \tag{8.2}$$

for a generalized problem of the form $Ax = \lambda Bx$. These methods fall under the general suggestive name shift-and-invert techniques. There are many possible ways of deriving efficient techniques based on shift and invert. In this section we will discuss some of the issues with one particular implementation in mind which involves a shift-and-invert preconditioning of Arnoldi's Algorithm.

1.1. General Concepts

Typically shift-and-invert techniques are combined with an efficient projection method such as Arnoldi's method or the Subspace iteration. The simplest possible scheme is to choose a shift σ and run Arnoldi's method on the matrix $(A - \sigma I)^{-1}$. Since the eigenvectors of A and $(A - \sigma I)^{-1}$ are identical one can recover the eigenvalues of A from the computed eigenvectors. Note that this can be viewed as an acceleration of the inverse iteration algorithm seen in Chapter IV, by Arnoldi's method, in the same way that the usual Arnoldi method was regarded as an acceleration of the power method. It requires only one factorization with the shifted matrix.

More elaborate algorithms involve selecting automatically new shifts and performing a few factorizations. Strategies for adaptively choosing new shifts and deciding when to refactor $(A - \sigma B)$ are usually referred to as shift-and-invert strategies. Thus, shift-and-invert simply consists of transforming the original problem

$(A - \lambda I)x = 0$ into $(A - \sigma I)^{-1}x = \mu x$. The transformed eigenvalues μ_i are usually far better separated than the original ones which results in better convergence in the projection type algorithms. However, there is a trade-off when using shift-and-invert, because the original matrix by vector multiplication which is usually inexpensive, is now replaced by the more complex solution of a linear system at every step. When a new shift σ is selected, the LU factorization of the matrix $(A - \sigma I)$ is performed and subsequently, at every step of Arnoldi's algorithm (or any other projection algorithm), an upper and a lower triangular systems are solved. Moreover, the cost of the initial factorization can be quite high and in the course of an eigenvalue calculation, several shifts, and therefore several factorizations, may be required. Despite these additional costs shift-and-invert is often an extremely useful technique, especially for generalized problems.

If the shift σ is suitably selected the matrix $C = (A - \sigma I)^{-1}$ will have a spectrum with much better separation properties than the original matrix A and therefore should require far less iterations to converge. Thus, the rationale behind the Shift-and-Invert technique is that factoring the matrix $(A - \sigma I)$ once, or a few times during a whole run in which σ is changed a few times, is a price worth paying because the number of iterations required with C is so much smaller than that required with A that the expense of the factorizations is amortized. For the symmetric generalized eigenvalue problem $Bx = \lambda Ax$ there are further compelling reasons for employing shift-and-invert. These reasons are well-known and have been discussed at length in the recent literature, see for example, [117, 118, 43, 160]. The most important of these is that since we must factor one of the matrices A or B in any case, there is little incentive in not factoring $(A - \sigma B)$ instead, to gain faster convergence. For this reason shift and invert has become a fairly standard tool in structural analysis because of the predominance of generalized eigenvalue problems in this application area.

For nonsymmetric eigenvalue problems, shift-and-invert strategies are not as well-known, although the main arguments supporting such techniques are the same as in the Hermitian case. Let us consider the case where the matrices B and A are real and banded but the shift σ is complex. One possibility is to work entirely in complex arithmetic. This is probably a fine alternative. If the matrix is real, it seems that the approach is a little wasteful and also unnatural. For example, it is known that the eigenvalues of the original matrix pencil come in complex conjugate pairs (at least in the case where B is positive definite). It would be desirable to have algorithms that deliver complex conjugate pairs as well. This is mainly because there may be a few close pairs of computed eigenvalues and it will become difficult to match the various pairs together if the conjugates are only approximately computed. A wrong match may in fact give incorrect eigenvectors. In the next section we consider the problem of performing the computations in real arithmetic.

1.2. Dealing with Complex Arithmetic

Let A be real and assume that we want to use a complex shift

$$\sigma = \rho + i\theta \ . \tag{8.3}$$

One can factor the matrix $(A - \sigma I)$ in (8.1) and proceed with an algorithm such as Arnoldi's method working with complex arithmetic. However, an alternative to using complex arithmetic is to replace the complex operator $(A - \sigma)^{-1}$ by the real one

$$B_+ = \Re e\left[(A - \sigma I)^{-1}\right] = \frac{1}{2}\left[(A - \sigma I)^{-1} + (A - \bar{\sigma} I)^{-1}\right] \tag{8.4}$$

whose eigenvectors are the same as those of the original problem and whose eigenvalues μ_i^+ are related to the eigenvalues λ_i of A by

$$\mu_i^+ = \frac{1}{2}\left(\frac{1}{\lambda_i - \sigma_i} + \frac{1}{\lambda_i - \bar{\sigma}_i}\right). \tag{8.5}$$

We can also use

$$B_- = \Im m\left[(A - \sigma I)^{-1}\right] = \frac{1}{2i}\left[(A - \sigma I)^{-1} - (A - \bar{\sigma} I)^{-1}\right] \ . \tag{8.6}$$

Again, the eigenvectors are the same as those of A and the eigenvalues μ_i^- are given by

$$\mu_i^- = \frac{1}{2i}\left(\frac{1}{\lambda_i - \sigma_i} + \frac{1}{\lambda_i - \bar{\sigma}_i}\right) . \tag{8.7}$$

A few additional possibilities are the following

$$B(\alpha, \beta) = \alpha B_+ + \beta B_- \ ,$$

for any nonzero pair α, β and

$$B_* = (A - \sigma I)^{-1}(A - \bar{\sigma})^{-1}. \tag{8.8}$$

This last option is known as the double shift approach and has been used by J.G.F. Francis in 1961/62 [48] in the context of the QR algorithm to solve a similar dilemma. The inverse of B_* is

$$(A - \sigma I)(A - \bar{\sigma} I) = [(A - \rho I)^2 + \theta^2 I].$$

This matrix, which is real, and is a quadratic polynomial in A and again shares A's eigenvectors. An interesting observation is that (8.8) is redundant with (8.6).

Proposition 8.1 *The matrices B_* and B_- are related by*

$$B_- = \theta B_* \ . \tag{8.9}$$

The proof is left as an exercise, see Exercise P-8.4.

An obvious advantage in using either (8.4) or (8.6) in place of (8.1) is that the first operator is real and therefore all the work done in the projection method can be performed in real arithmetic. A nonnegligible additional benefit is that the complex conjugate pairs of eigenvalues of original problem are also approximated by complex conjugate pairs thus removing some potential difficulties in distinguishing these pairs when they are very close. In a practical implementation, the matrix $(A - \sigma I)$ must be factored into the product LU of a lower triangular matrix L and an upper triangular matrix U. Then every time the vector $w = \Re e[(A - \sigma I)^{-1}]v$ must be computed, the forward and backward solves are processed in the usual way, possibly using complex arithmetic, and then the real part of the resulting vector is taken to yield w.

An interesting question that might be asked is which of (8.4) or (8.6) is best? The experiments in [123] reveal that the choice is not an easy one. It is readily verified that as $\lambda \to \sigma$,

$$\mu^+ \approx \frac{1}{2(\lambda - \sigma)} , \quad \mu^- \approx \frac{1}{2i(\lambda - \sigma)}.$$

indicating that B_+ and B_- give the same enhancement to eigenvalues close to σ. In contrast, as $\lambda \to \infty$, B_- dampens the eigenvalues more strongly than does B_+ since,

$$\mu^+ = \frac{\lambda - \rho}{(\lambda - \sigma)(\lambda - \bar{\sigma})} , \quad \mu^- = \frac{\theta}{(\lambda - \sigma)(\lambda - \bar{\sigma})} . \tag{8.10}$$

The only conclusion from all this is that whichever of the two options is used the performance is not likely to be substantially different from the other or from that of the standard (8.1).

In the following discussion we choose to single out B_+, but all that is said about B_+ is also true of B_-. In practice it is clear that the matrix B_+ should not be computed explicitly. In fact either of these matrices is full in general and would be prohibitive to compute. Instead, we first factor the matrix $(A - \sigma I)$ at the outset. This is done in complex arithmetic or by implementing complex arithmetic with real arithmetic. For example, if A is banded, to preserve bandedness and still use real arithmetic, one can represent the j-th component $x_j = \xi_j + i\zeta_j$ of a vector z of $\mathbb{C}^n$ by the components $\eta_{2j-1} = \xi_j$ and $\eta_{2j} = \zeta_j$ of the real 2n-vector y of the components η_j, $j = 1, \ldots, 2n$. Translating the matrix $(A - \sigma I)$ into this transformation gives a $(2n) \times (2n)$ real banded matrix. Once the matrix is factored, a projection type method, e.g., subspace iteration, is applied using as operator $B_+ = \Re e(A - \sigma I)$. Matrix-vector products with the matrix B_+ are required in the subspace iteration. Each of these can be performed as follows.

1. Solve $(A - \sigma I)w = v$ (possibly in complex arithmetic).

2. Set $B_+v = \Re e(w)$ (respectively $B_-v = \Im m(w)$).

1.3. Shift-and-Invert Arnoldi

We now consider the implementation of shift-and-invert with an algorithm such as Arnoldi's method. Assume that the problem is to compute the p eigenvalues closest to a shift σ_0. In the symmetric case there is an important tool that is used to determine which of the approximate eigenvalues should be considered in order to be able to compute all the desired eigenvalues in a given interval only once. This tool is Sylvester's inertia theorem which gives the number of eigenvalues to the right and left of σ by counting the number of negative entries in the diagonal elements of the U part of the LU factorization of the shifted matrix. In the non Hermitian case a similar tool does not exist. In order to avoid the difficulty we exploit deflation in the following manner. As soon as an approximate eigenvalue has been declared satisfactory we proceed to a deflation process with the corresponding Schur vector. The next run of Arnoldi's method will attempt to compute some other eigenvalue close to σ_0. With proper implementation, the next eigenvalue will usually be the next closest eigenvalue to σ_0. However, there is no guarantee for this and there is no guarantee that an eigenvalue will not be missed. This is a weakness of projection methods in the non Hermitian case, in general.

Our experimental code ARNINV based on this approach implements a simple strategy which requires two parameters m, k_{rest} from the user and proceeds as follows. The code starts by using σ_0 as an initial shift and calls Arnoldi's algorithm with $(A - \sigma_0 I)^{-1}$ Arnoldi to compute the eigenvalue of A closest to σ_0. Arnoldi's method is used with restarting, i.e., if an eigenvalue fails to converge after the Arnoldi loop we rerun Arnoldi's algorithm with the initial vector replaced by the eigenvalue associated with the eigenvalue closest to σ_0. The strategy for changing the shift is dictated by the second parameter k_{rest}. If after k_{rest} calls to Arnoldi with the shift σ_0 the eigenpair has not yet converged then the shift σ_0 is changed to the best possible eigenvalue close to σ_0 and we repeat the process. As soon as the eigenvalue has converged we deflate it using Schur deflation as described in the previous section. The algorithm can be summarized as follows.

ALGORITHM **8.1 Shift-and-Invert Arnoldi**

1. **Initialize:**

   Choose an initial vector v_1 of norm unity, an initial shift σ, and the dimension and restart parameters m and k_{rest}.

2. **Eigenvalue loop:**

   (a) Compute the LU factorization of $(A - \sigma I)$.

   (b) If $k > 1$ then (re)-compute

   $$h_{ij} = ((A - \sigma I)^{-1} v_j, v_i) \;\; i, j = 1, k-1 \; .$$

   (c) Arnoldi Iteration. For $j = k, k+1, ..., m$ do:

   - *Compute $w := (A - \sigma I)^{-1} v_j$.*
   - *Compute a set of j coefficients h_{ij} so that $w := w - \sum_{i=1}^{j} h_{ij} v_i$ is orthogonal to all previous v_i's, $i = 1, 2, ..., j$.*
   - *Compute $h_{j+1,j} := \|w\|_2$ and $v_{j+1} := w / h_{j+1,j}$.*

   (d) Compute eigenvalue of H_m of largest modulus, corresponding approximate eigenvector of $(A - \sigma I)^{-1}$, and associated (estimated) residual norm ρ_k.

   (e) Orthonormalize this eigenvector against all previous v_j's to get the approximate Schur vector $\tilde{u}_k$ and define $v_k := \tilde{u}_k$.

(f) *If ρ_k is small enough then accept v_k as the next Schur vector. Set $k : k + 1$; if $k<p$ goto 2.*

(g) *If the number of restarts with the same shift exceeds k_{rest} select a new shift and goto 1. Else restart Arnoldi's algorithm, i.e., goto 2-(c).*

A point of detail in the algorithm is that the $(k-1) \times (k-1)$ principal submatrix of the Hessenberg matrix H_m is recomputed whenever the shift changes. The reason is that this submatrix represents the matrix $(A - \sigma I)^{-1}$ in the first $k-1$ Schur vectors and therefore it must be updated as σ changes. This is in contrast with the simple Arnoldi procedure with deflation described earlier in Chapter VI. However, there exists a simpler implementation that avoids this, see Exercise P-8.2. The above algorithm is described for general complex matrix and there is no attempt in it to avoid complex arithmetic in case the original matrix is real. In this situation, we must replace $(A - \sigma I)^{-1} v_j$ in B.2 by $\Re e[(A - \sigma I)^{-1} v_j]$ and ensure that we select the eigenvalues corresponding to the eigenvalues of A closest to σ. We also need to replace the occurrences of eigenvectors by the pair of real parts and imaginary parts of the eigenvectors.

Example 8.1 We consider the test problem on Chemical reactions described in Chapter III. This coupled system is discretized in the interval $[0, 1]$ using $n_x + 1$ points with $n_x = 100$ which yields a matrix of size $n = 200$. We tested ARNINV to compute the six rightmost eigenvalues of A. We took as initial shift the value $\sigma = 0$, and $m = 15$, $k_{rest} = 10$. In this case ARNINV delivered all the desired eigenvalues by making four calls to the Arnoldi subroutine and there was no need to change shifts. The tolerance imposed was $\epsilon = 10^{-7}$. The result of the execution is shown in Table 8.1. What is shown in the figure is the progress of the algorithm after each projection (Arnoldi) step. The eigenvalue loop number indicates the eigenvalue that is being computed at the particular Arnoldi call. Thus, when trying to compute the eigenvalue number 3, the algorithm has already computed the first two (in this case a complex conjugate pair), and has deflated them. We print the eigenvalue of interest, i.e., the one we are trying to compute, plus the one (or the pair of complex conjugate eigenvalues) that is likely to converge after it. The last column shows the actual residual norm achieved for the eigenvalues shown. After execution, we computed the average error for the 6 computed eigenvalues and found that it was equal to 0.68×10^{-14}. The total execution time on an Alliant FX-8 computer was about 2.13 seconds.

Eig.	$\Re e(\lambda)$	$\Im m(\lambda)$	Res. Norm
1	0.1807540453D-04	0.2139497548D+01	0.212D-09
	0.1807540453D-04	-0.2139497548D+01	0.212D-09
	-0.6747097569D+00	0.2528559918D+01	0.224D-06
	-0.6747097569D+00	-0.2528559918D+01	0.224D-06
3	-0.6747097569D+00	0.2528559918D+01	0.479D-13
	-0.6747097569D+00	-0.2528559918D+01	0.479D-13
	-0.2780085122D+01	0.2960250300D+01	0.336D-01
	-0.2780085122D+01	-0.2960250300D+01	0.336D-01
5	-0.1798530837D+01	0.3032164644D+01	0.190D-06
	-0.1798530837D+01	-0.3032164644D+01	0.190D-06
5	-0.1798530837D+01	0.3032164644D+01	0.102D-11
	-0.1798530837D+01	-0.3032164644D+01	0.102D-11
	-0.2119505960D+02	0.1025421954D+00	0.749D-03

Table 8.1 Convergence history of ARNINV for chemical reaction test problem. Each separate outer iteration corresponds to a call to Arnoldi's module

We rerun the above test with a larger number of eigenvalues to compute, namely $nev = 10$. The initial shift σ, was changed to $\sigma_0 = -0.5 + 0.2i$ and we also changed k_{rest} to $k_{rest} = 3$. Initially, the run looked similar to the previous one. A pair of complex conjugate eigenvalues were found in the first Arnoldi iteration, then another pair in the second iteration, then none in the third iteration and one pair in the fourth iteration. It took two more iterations to get the eigenvalues number 7 and 8. For the last eigenvalue a new shift was taken because it took three Arnoldi iterations without success. However the next shift that was taken was already an excellent approximation and the next eigenvalue was computed in the next iteration. The cost was higher than the previous run with the CPU time on the Alliant FX-8 climbing to approximately 5.65 seconds.

2. Polynomial Preconditioning

We have seen in the previous chapter a few different ways of exploiting polynomials in A to accelerate simple algorithms such as Arnoldi's method or subspace iteration. In this section we will show another way of combining a projection type technique such as Arnoldi's method with these polynomials.

For a classical eigenvalue problem, one alternative is to use polynomial preconditioning as is described next. The idea of polynomial preconditioning is to replace the operator B by a simpler matrix provided by a polynomial in A. Specifically, we consider the polynomial in A

$$B_k = p_k(A) \tag{8.11}$$

where p_k is a degree k polynomial. Ruhe [135] considers a more general method in which p_k is not restricted to be a polynomial but can be a rational function. When an Arnoldi type method is applied to B_k, we do not need to form B_k explicitly, since all we will ever need in order to multiply a vector x by the matrix B_k is k matrix-vector products with the original matrix A and some linear combinations.

For fast convergence, we would ideally like that the r wanted eigenvalues of largest real parts of A be transformed by p_k into r eigenvalues of B_k that are very large as compared with the remaining eigenvalues. Thus, we can proceed as in the previous chapter by attempting to minimize some norm of p_k in some region subject to constraints of the form,

$$p(\lambda_1) = 1 \quad \text{or} \quad \sum_{j=1}^{r} \mu_j p(\lambda_j) = 1 \ . \tag{8.12}$$

Once again we have freedom in choosing the norm of the polynomials, to be either the infinity norm or the L_2-norm. Because the L_2-norm offers more flexibility and performs usually slightly better than the infinity norm, we will only consider a technique based on the least squares approach. We should emphasize, however, that a similar technique using Chebyshev polynomials can easily be developed. Therefore, we are faced again with the function approximation problem described in Section 3.3.

Once p_k is calculated, the preconditioned Arnoldi process consists of using Arnoldi's method with the matrix A replaced by $B_k = p_k(A)$. This will provide us with approximations to the eigenvalues of B_k which are related to those of A by $\lambda_i(B_k) = p_k(\lambda_i(A))$ It is clear that the approximate eigenvalues of A can be obtained from the computed eigenvalues of B_k by solving a polynomial equation. However, the process is complicated by the fact that there are k roots of this equation for each value $\lambda_i(B_k)$ that are candidates for representing one eigenvalue $\lambda_i(A)$. The difficulty is by no means unsurmountable but we have preferred a more expensive but simpler alternative based on the fact that the eigenvectors of A and B_k are identical. At the end of the Arnoldi process we obtain an orthonormal basis V_m which contains all the approximations to these eigenvectors. A simple idea is to perform a Galerkin process for A onto span$[V_m]$ by explicitly computing the matrix $A_m = V_m^H A V_m$ and its eigenvalues and eigenvectors. Then the approximate eigenvalues of A are the eigenvalues of A_m and the approximate eigenvectors are given by $V_m y_i^{(m)}$ where $y_i^{(m)}$ is an eigenvector of A_m associated with the eigenvalue $\tilde{\lambda}_i$. A sketch of the algorithm for computing nev eigenvalues is as follows.

ALGORITHM **8.2 Least-Squares Preconditioned Arnoldi**

1. **Start:** *Choose the degree k of the polynomial p_k, the dimension parameter m and an initial vector v_1. Set $iev = 1$.*

2. **Initial Arnoldi Step:** *Using the initial vector v_1, perform m steps of the Arnoldi method with the matrix A and get initial set of Ritz values for A.*

3. **Eigenvalue Loop:**

(a) *Adapt:* *From the previous convex hull and the new set of Ritz values construct a new convex hull of the unwanted eigenvalues. Obtain the new least squares polynomial* p_k *of degree* k.

(b) *Update* H_m: *If* $iev > 1$ *then (re)-compute*

$$h_{ij} = (p_k(A)v_j, v_i) \;\; i, j = 1, iev - 1 \; .$$

(c) *Arnoldi Iteration:* *For* $j = iev, iev + 1, ..., m$ *do:*

- *Compute* $w := p_k(A)v_j$
- *Compute a set of* j *coefficients* h_{ij} *so that* $w := w - \sum_{i=1}^{j} h_{ij} v_i$ *is orthogonal to all previous* v_i*'s,* $i = 1, 2, ..., j$.
- *Compute* $h_{j+1,j} := \|w\|_2$ *and* $v_{j+1} := w/h_{j+1,j}$.

(d) *Projection Step:* *Compute the matrix* $A_m = V_m^T A V_m$ *and its* m *eigenvalues* $\{\tilde{\lambda}_1, \ldots \tilde{\lambda}_m\}$.

(e) *Select* *the next wanted approximate eigenvalue* $\tilde{\lambda}_{iev}$ *and compute corresponding eigenvector* $\tilde{z}$. *Orthonormalize this eigenvector against* $v_1, \ldots, v_{iev-1}$ *to get the approximate Schur vector* $\tilde{u}_{iev}$ *and define* $v_{iev} := \tilde{u}_{iev}$.

(f) *Test*. *If* ρ_{iev} *is small enough then accept* v_{iev} *as the next Schur vector and set* $iev := iev + 1$.

(g) *Restart:* *if* $iev<nev$ *goto 2.*

The general structure of the algorithm is quite close to that of shift-and-invert with deflation. What differentiates the two algorithms is essentially the fact that here we need to adaptively compute a polynomial, while the shift-and-invert algorithm computes an LU factorization of a shifted matrix. Practically, we must be careful about the number of factorizations needed in shift-and-invert whereas the computational cost of calculating a new polynomial is rather low. The difference between this method and those of the previous chapter is that here the polynomial iteration is an inner iteration and the Arnoldi iteration is the outer loop, while in the hybrid method, the two processes are serially following each other. Both approaches can be viewed as means of accelerating the Arnoldi method.

It is clear that a version without the Schur-Wielandt deflation technique can also be applied to the polynomial preconditioned Arnoldi method but this is not recommended.

Example 8.2 We take the same example as in the previous section and illustrate the use of an experimental least squares Arnoldi program called ARNLS on the above example. We fixed the dimension of the Krylov subspace to be always equal to $m = 15$. The degree of the polynomial was taken to be 20. However, note that the program has the capability to lower the degree by as much as is required to ensure a well conditioned Gram matrix in the least squares polynomial problem. This did not happen in this run however, i.e. the degree was always 20. Again, ARNLS was asked to compute the six rightmost eigenvalues. The run was much longer so its history cannot be reproduced here. Here are however a few statistics.

- Total number of matrix by vector multiplications for the run = 2053;
- Number of calls to the projection subroutines = 9;
- Total CPU time used on an Alliant FX-8 = 3.88 sec.

Note that the number of projection steps is more than twice that required for shift-and-invert. The execution time is also more than 80 % higher. We rerun the same program by changing only two parameters: m was increased to $m = 20$ and the degree of the polynomial was set to $k = 15$. The statistics are now as follows:

- Total number of matrix by vector multiplications for the run = 1144;
- Number of calls to the projection subroutines = 5;
- Total CPU time used = 2.47 sec.

Both the number of projection steps and the execution times have been drastically reduced and have come closer to those obtained with shift-and-invert.

One of the disadvantages of polynomial preconditionings is precisely this wide variation in performance depending on the choice of the parameters. To some extent there is a similar dependence of the performance of ARNINV on the initial shift, although in practice a good initial shift is often known. A superior feature of shift-and-invert is that it allows to compute eigenvalues inside the spectrum. Polynomial preconditioning can be generalized to this case but does not perform too well. We should also comment on the usefulness of using polynomial preconditioning in general. A commonly heard argument against polynomial preconditioning is that is it suboptimal: In the Hermitian case the conjugate gradient and the Lanczos methods are optimal polynomial processes in that they provide the best possible approximation, in some sense, to the original problem from Krylov subspaces. Hence the argument that polynomial preconditioning would not perform as well since it si likely to require a larger number of matrix by vector multiplications. However, in the non Hermitian case the optimality result is no longer valid. In fact even in the symmetric case the optimality result is only true in exact arithmetic, which is far from real situations in which loss of orthogonality can be rather severe. A notable difference with the situation of linear system solutions is that the overhead in computing the best ellipse and best polynomial may now be amortized over several eigenvalues. In fact one single outer loop may enable one to compute a few eigenvalues/eigenvectors and not just one.

The next question is whether or not a simple restarted Arnoldi algorithm would perform better than a polynomial preconditioned method. The answer is a definite no. A run with ARNIT [148] an iterative Arnoldi method with deflation failed even to deliver the first eigenvalue of the test matrix used in the above example. The initial vector was the same and we tried two cases $m = 15$, which did not show any sign of convergence and $m = 20$ which might have eventually converged but was extremely slow. The nonrestarted Arnoldi method would, however be of interest, if not for its excessive memory requirement.

3. Davidson's Method

Davidson's method is a generalization of the Lanczos algorithm in that like the Lanczos algorithm it uses projections of the matrix over a sequence of subspaces of increasing dimension. It is indeed a preconditioned version of the Lanczos method. The difference with the Lanczos algorithm is that the amount of work required at each step increases at each iteration because, just like in Arnoldi's method, we must orthogonalize against all previous vectors. From the implementation point of view the method is akin to Arnoldi's method. For example, the process must be restarted periodically with the current best estimate of the wanted eigenvector.

The basic idea of the algorithm is rather simple. It consists of generating an orthogonal set of vectors onto which a projection is performed. At each step j, (this is the equivalent to the j-th step in the Lanczos algorithm) the residual vector of the current approximation $\tilde{\lambda}, \tilde{u}$ to the desired eigenpair is computed. The resulting vector is then multiplied by $(M - \tilde{\lambda} I)^{-1}$, where M is some preconditioning matrix. In the original algorithms M was simply the diagonal of the matrix A.

Thus, the algorithm consists of two nested loops. The process for computing the largest (resp. smallest) eigenvalue of A, can be described as follows.

ALGORITHM **8.3 Davidson's method.**

1. **Start:** *Choose an initial unit vector* v_1.

2. **Iterate:** *Until convergence do:*

3. **Inner Loop:** *for* $j = 1, \ldots, m$ *do:*

   - *Compute* $w := Av_j$.
   - *Compute* $V_j^T w$, *the last column of* $H_j := V_j^T A V_j$.
   - *Compute the largest eigenpair* λ, y *of* H_j.

- *Compute the Ritz vector $u := V_j y$ and its associated residual vector $r := Au - \lambda u$.*
- *Test for convergence. If satisfied Return.*
- *Compute $t := M_j r$ (skip when $j = m$).*
- *Orthogonalize t against V_j via Modified Gram-Schmidt: $V_{j+1} := MGS([V_j, t])$ (skip when $j = m$).*

4. **Restart:** *Set $v_1 := u$ and go to 3.*

The preconditioning matrix M_j is normally some approximation of $(A - \lambda I)^{-1}$. As was already mentioned the simplest and most common preconditioner M_j is $(D - \lambda I)^{-1}$ where D is the main diagonal of A (Jacobi Preconditioner). It can only be effective when A is nearly diagonal, i.e., when matrix of eigenvectors is close to the identity matrix. The fact that this is often the situation in Quantum Chemistry explains the popularity of the method in this field. However, the preconditioner need not be as simple. It should be noticed that, without preconditioning, i.e., when if $M_j = I$ for all j, then the sequence of vectors v_j coincide with those produced by the Lanczos algorithm, so that the Lanczos and Davidson algorithms are equivalent in this case.

When several eigenvalues are sought or when it is known that there is a cluster of eigenvalues around the desired eigenvalue then a block version of the algorithm may be preferred. Then several eigenpairs of H_j will be computed at the same time and several vectors are added to the basis V_j instead of one.

We state a general convergence result due to Sadkane [153]. In the following, we assume that we are seeking to compute the largest eigenvalue λ_1. We denote by $\mathcal{P}_j$ the projection onto a subspace K_j spanned by an orthonormal basis V_j. Thus, the *nonrestarted* Davidson algorithm is just a particular case of this situation.

Theorem 8.1 *Assuming that the Ritz vector $u_1^{(j)}$ belongs to K_{j+1}, then the sequence of Ritz values $\lambda_1^{(j)}$ is an increasing sequence that* *is convergent. If, in addition, the preconditioning matrices are uniformly bounded and uniformly positive definite in the orthogonal complement of K_j and if the vector $(I - \mathcal{P}_j)M_j r_j$ belongs to K_{j+1} for all j then the limit of $\lambda_1^{(j)}$ as $j \to \infty$ is an eigenvalue of A and $u_1^{(j)}$ admits a subsequence that converges to an associated eigenvector.*

Proof. For convenience the subscript 1 is dropped from this proof. In addition we assume that all matrices are *real* symmetric. That $\lambda^{(j)}$ is an increasing sequence is a consequence of the assumptions and the min-max theorem. In addition, the $\lambda^{(j)}$ is bounded from above by λ and as result it converges.

To prove the second part of the theorem, let us define $z_j = (I - \mathcal{P}_j)M_j r_j$ and $w_j = z_j/\|z_j\|_2$. Note that since $u^{(j)} \perp z_j$ and $r_j \perp K_j$ we have,

$$\begin{aligned} z_j^H A u^{(j)} &= z_j^H(\lambda^{(j)} u^{(j)} + r_j) \\ &= r_j^H M_j (I - \mathcal{P}_j) r_j \\ &= r_j^H M_j r_j. \end{aligned} \tag{8.13}$$

Consider the 2-column matrix $W_j = [u^{(j)}, w_j]$ and let

$$B_j = W_j^H A W_j = \begin{pmatrix} \lambda^{(j)} & \alpha_j \\ \alpha_j & \beta_j \end{pmatrix} \tag{8.14}$$

in which we have set $\alpha_j = w_j^H A u^{(j)}$ and $\beta_j = w_j^H A w_j$. Note that by the assumptions span$\{W_j\}$ is a subspace of K_{j+1}. Therefore, by Cauchy's interlace theorem and the optimality properties of the Rayleigh Ritz procedure the smallest of two eigenvalues $\mu_1^{(j)}, \mu_2^{(j)}$ of B_j satisfies the relation

$$\lambda^{(j)} \le \mu_1^{(j)} \le \lambda^{(j+1)}.$$

The eigenvalues of B_j are defined by $(\mu - \lambda^{(j)})(\mu - \beta_j) - \alpha_j^2 = 0$ and as a result of $|\mu_1^{(j)}| \le \|A\|_2$ and $|\beta_j| \le \|A\|_2$ we

$$\alpha_j^2 \le 2(\mu_1^{(j)} - \lambda^{(j)})\|A\|_2 \le (\lambda^{(j+1)} - \lambda^{(j)})\|A\|_2 \ .$$

The right hand side of the above inequality converges to zero as $j \to \infty$ and so $\lim_{j\to\infty} = 0$. From (8.13),

$$r_j^H M_j r_j = \|z_j\|_2 \alpha_j \leq \|(I - \mathcal{P}_j) M_j r_j\| \alpha_j \leq \|M_j r_j\| \alpha_j \ .$$

Since we assume that M_j is uniformly bounded and using the fact that $\|r_j\|_2 \leq 2\|A\|_2$ the above inequality shows that

$$\lim_{j\to\infty} r_j^H M_j r_j = 0.$$

In addition, since r_j belongs to the orthogonal complement of K_j and by the uniform positive definiteness of the sequence M_j, $r_j^H M_j r_j \geq \gamma \|r_j\|_2{}^2$ where γ is some positive constant. Therefore, $\lim_{j\to\infty} r_j = 0$. To complete the proof, let $\bar{\lambda}$ the limit of the sequence $\lambda^{(j)}$. The $u^{(j)}$'s are bounded since they are all of norm unity so they admit a limit point. Taking the relation $r_j = (A - \lambda^{(j)} I) u^{(j)}$, to the limit, we see that any such limit point $\bar{u}$, must satisfy $(A - \bar{\lambda} I)\bar{u} = 0$. ∎

The result given by Sadkane includes the more general case where more than one eigenvalue is computed by the algorithm and is therefore more general, see Exercise P-8.1 for details. The restriction on the positive definiteness of the M_j 's is a rather severe condition in the case where the eigenvalue to be computed is not the largest one. The fact that M_j must remain bounded is somewhat less restrictive. However, in shift-and-invert preconditioning, for example, an unbounded M_j is sought rather than avoided. If we want to achieve rapid convergence, it is desirable to have M_j close to some $(A - \sigma I)^{-1}$ in some sense and σ close to the desired eigenvalue. The assumptions of the theorem do not allow us to take σ too close from the desired eigenvalue. Nevertheless, this result does establish convergence in some instances and we should add that little else is known concerning the convergence of Davidson's algorithm.

4. Generalized Arnoldi Algorithms

It is interesting to note that the generalized Davidson methods are similar to preconditioned conjugate gradient type methods. The only additional feature is that the preconditioning is allowed to vary at each step. We can define a preconditioned Arnoldi procedure using similar ideas. The subspace is constructed by adding at each step a new vector of the form

$$AM_j^{-1}r_j$$

which is then orthonormalized against $v_1, \ldots v_j$ to yield v_{j+1}. In the above equation M_j is the preconditioner, which in the original Davidson method is defined as

$$M_j = \text{diag}\,(A) - \tilde{\lambda}_j I$$

where $\tilde{\lambda}_j$ is the current approximation to the desired eigenvalue. In the symmetric case the usual implementation of Davidson's method requires computing the matrix

$$C_j = V_j^H A V_j$$

which is updated at every step. Because of symmetry of the matrix C_j this necessitates the computation of exactly j inner products, namely (Av_j, v_i), $i = 1, 2, \ldots j$, at step j.

In contrast, the non-Hermitian case does not allow such a simple updating mechanism. The simplest possibility in this situation would be to save the two sets of vectors v_j and $w_j \equiv Av_j$ generated at every step, thus essentially doubling memory requirement. The matrix $C_j = V_j^H W_j$ can then be updated at each step, where $V_j = [v_1, v_2, ..., v_j]$, $W_j = [w_1, w_2, ..., w_j]$. An alternative is to compute (Av_j, v_i), $i \le j$ as before and $h_{ji} = (Av_i, v_j)$ $i<j$ via

$$h_{ji} = (v_i, A^H v_j)$$

which requires another matrix – vector product but does not necessitate saving the w_j's. This allows us to add row j and column

j of the matrix C_m at step j. In spite of these unattractive additional costs, this technique is appealing because of its exceptional flexibility. Any preconditioner can be used and it is allowed to vary at every step. For example, in polynomial preconditioning a different polynomial can be used at each step. Similarly, in Shift-and-Invert the shift can be changed at any step.

A particularly important application of this technique is when computing eigenvalues with largest real parts of a large matrix. In this situation it is desirable to use a preconditioning that computes an approximation to $\exp(A)v_j$, i.e., at step j of the preconditioned procedure we would like to have

$$M_j v_j \approx \exp(A) v_j.$$

Any such approximation can be used. For example, a technique based on Krylov subspace approximations developed in [53, 152] is suitable. The approximation is of the form $\exp(A)v \approx q_m(A)v$, where q_m is a polynomial of degree $m-1$ that depends on the vector v. In fact this can be viewed as a conjugate-gradient type algorithm for approximating the exponential propagation operator. The framework of the preconditioned Arnoldi algorithm described above seems perfectly suitable for incorporating these variable preconditioners.

PROBLEMS

P-8.1 Consider a more general framework for Theorem 8.1, in which one is seeking l eigenvalues at a time. The new vectors are defined as

$$t_{i,j} = M_{i,j}^{-1} r_{i,j} \quad i = 1, 2, \ldots l.$$

where i refers to the eigenvalue number and j is the step number. As a result the dimension of the Davidson subspace increases by l at every step instead of one. The assumptions on each of the individual preconditioners for each eigenvalue are the same as in Theorem 8.1. (1) Establish that the first part of Theorem 8.1 is still valid.

(2) To show the second part, define $z_{ij} = (I - \mathcal{P}_j)M_{i,j}r_{ij}$ and similarly $w_{ij} = z_{ij}/\|z_{ij}\|_2$ and

$$W_j = [u_1^{(j)}, u_2^{(j)}, \ldots, u_i^{(j)}, w_{ij}].$$

Show that W_j is orthonormal and that the matrix $B_{i,j} = W_j^H A W_j$ has the form,

$$B_{i,j} = \begin{pmatrix} \lambda_1^{(j)} & & & \alpha_{1j} \\ & \ddots & & \vdots \\ & & \lambda_i^{(j)} & \alpha_{ij} \\ \alpha_{1j} & \cdots & \alpha_{ij} & \beta_j \end{pmatrix} \tag{8.15}$$

in which we set $\alpha_{kj} = w_{ij}^H A u_k^{(j)})$ and $\beta_j = w_{ij}^H A w_{ij}$.
(3) Show that

$$\lambda_k^{(j)} \le \mu_k^{(j)} \le \lambda_k^{(j+1)} \quad k = 1, 2, \ldots, i.$$

(4) Taking the Frobenius norm of $B_{i,j}$ and using the fact that

$$\mathrm{tr}(B_{i,j}) = \sum_{k=1}^{i+1} \mu_k^{(j)} = \beta_j + \sum_{k=1}^{i} \lambda_k^{(j)}$$

show that

$$\begin{aligned} 2\sum_{k=1}^{i} \alpha_{kj}^2 &= \sum_{k=1}^{i} (\mu_k^{(j)} - \lambda_k^{(j)})(\mu_k^{(j)} + \lambda_k^{(j)} - \mu_{i+1}^{(j)} - \beta_j) \\ &\le 4\|A\|_2 \sum_{k=1}^{i} (\mu_k^{(j)} - \lambda_k^{(j)}) \ . \end{aligned}$$

(5) Complete the proof of the result similarly to Theorem 8.1.

P-8.2 Using the result of Exercise P-6.3 write a simpler version of the shift-and-invert Arnoldi Algorithm with deflation, Algorithm 8.1, which does not require the $(k-1) \times (k-1)$ principal submatrix of H_m, i.e., the (quasi) upper triangular matrix representing of $(A - \sigma I)^{-1}$ in the computed invariant subspace.

P-8.3 How can one get the eigenvalues of A from those of B_+ or B_-. What happens if the approximate eigenvalues are close and complex? What alternative can you suggest for recovering approximate eigenvalues of A from a given projection process applied to either of these two real operators.

P-8.4 Establish the relation (8.9).

NOTES AND REFERENCES. Although the notion of preconditioning is well-known for linear systems it is not clear who defined this notion first. In the survey paper by Golub and O'Leary [60] it is stated that "The term *preconditioning* is used by Turing (1948) and by then seems standard terminology for problem transforming in order to make solutions easier. The first application of the work to the idea of improving the convergence of an iterative method may be by Evans (1968), and Evans (1973) and Axelsson (1974) apply it to the conjugate gradient algorithm". However, the idea of polynomial preconditioning is clearly described in a 1952 paper by Lanczos [90], although Lanczos does not use the term "preconditioning" explicitly. The idea was suggested later for eigenvalue calculations by Stiefel who employed least-squares polynomials [173] and Rutishauser [136] who combined the QD algorithm with Chebyshev acceleration. The section on Shift-and-Invert preconditioning is adapted from [123]. Davidson's method as described in [30] can be viewed as a cheap version of Shift-and-Invert , in which the solution of the linear systems are solved (very) inaccurately. The method is well-known to the physicists or quantum chemists but not as well known to numerical analysts. The lack of theory of the method might have been one reason for the neglect. Generalizations and extensions of the method are proposed by Morgan and Scott [104] in the Hermitian case but little has been done in the non-Hermitian case. ♠

Chapter IX

Non-Standard Eigenvalue Problems

Many problems arising in applications are not of the standard form $Ax = \lambda x$ but of the 'generalized' form $Ax = \lambda Bx$. In structural engineering, the A matrix is called the stiffness matrix and B is the mass matrix. In this situation, both are symmetric real and often B is positive definite. Other problems are quadratic in nature, i.e., they take the form

$$\lambda^2 Ax + \lambda Bx + Cx = 0.$$

This chapter gives a brief overview of these problems and of some specific techniques that are used to solve them. In many cases, we will seek to convert a nonstandard problems into a standard one in order to be able to exploit the methods and tools of the previous chapters.

1. Introduction

Many eigenvalue problems arising in applications are either generalized, i.e., of the form

$$Ax = \lambda Bx \tag{9.1}$$

or quadratic,

$$\lambda^2 Ax + \lambda Bx + Cx = 0.$$

Such problems can often be reduced to the standard form $Ax = \lambda x$ under a few mild assumptions. For example when B is nonsingular, then (9.1) can be rewritten as

$$B^{-1}Ax = \lambda x \ . \tag{9.2}$$

As will be explained later, the matrix $C = B^{-1}A$ need not be computed explicitly in order to solve the problem. Similarly, the quadratic eigen-problem can be transformed into a generalized eigen-problem of size $2n$, in a number of different ways.

Thus, it might appear that these nonstandard problems may be regarded as particular cases of the standard problems and that no further discussion is warranted. This is not the case. First, a number of special strategies and techniques must be applied to improve efficiency. For example, when A is symmetric and B is symmetric positive definite then an alternative transformation of (9.1) will lead to a Hermitian problem. Second, there are some specific issues that arise, such as the situation where both A and B are singular matrices, which have no equivalent in the standard eigenvalue context.

2. Generalized Eigenvalue Problems

In this section we will summarize some of the results known for the generalized eigenvalue problem and describe ways of transforming it into standard form. We will then see how to adapt some of the techniques seen in previous chapters.

2.1. General Results

The pair of matrices A, B in the problem (9.1) is often referred to as a *matrix pencil.* We will use both the terms *matrix pair* or matrix pencil. If there is no particular reason why one of the two matrices A and B should play a special role, then the most natural way of defining eigenvalues of a matrix pencil is to think of them as pairs (α, β) of complex numbers. Thus, (α, β) is an eigenvalue of the pair (A, B) if by definition there is a vector u, called an associated eigenvector, such that

$$\beta A u = \alpha B u. \tag{9.3}$$

Equivalently, (α, β) is an eigenvalue if and only if

$$\det(\beta A - \alpha B) = 0 \ .$$

When (α, β) is an eigenvalue pair for (A, B), then $(\bar{\alpha}, \bar{\beta})$ is an eigenvalue pair for (A^H, B^H) since $\det(\beta A - \alpha B)^H = 0$. The left eigenvector for A, B is defined as a vector for which

$$(\beta A - \alpha B)^H w = 0. \tag{9.4}$$

This extension of the notion of eigenvalue is not without a few drawbacks. First, we note that the trivial pair $(0, 0)$ always satisfies the definition. Another difficulty is that there are infinitely many pairs (α, β) which can be termed 'generalized eigenvalues' to represent the same 'standard eigenvalue'. This is because we can multiply a given (α, β) by any complex scalar and still get an eigenvalue for the pencil. Thus, the standard definition of an eigenvalue corresponds to the case where $B = I$ and $\beta = 1$. There are three known ways out of the difficulty. A popular way is to take the ratio α/β as an eigenvalue, which corresponds to selecting the particular pair $(\alpha, 1)$ in the set. When β is zero, the eigenvalue takes the value infinity and this may not be satisfactory from the numerical point of view. A second way would be to use pairs (α, β) but scale them by some norm in $\mathbb{C}^2$, e.g., so that $|\alpha|^2 + |\beta|^2 = 1$. Finally, a third way, adopted by Stewart and Sun [172] is to denote by $< \alpha, \beta >$ the set of all pairs that satisfy (9.3). The eigenvalue is then a set instead of an element in $\mathbb{C}^2$. We will refer to this set as a *generalized eigenvalue*. However, we will sacrifice a little of rigor for convenience, and also call any representative element (α, β), in the set, at the exclusion of $(0, 0)$, an *eigenvalue pair* . Note the distinction between the notation of an eigenvalue pair $(.,.)$ and the set to which it belongs to, i.e., the generalized eigenvalue, denoted by $< .,. >$. This definition is certainly radically different from, and admittedly more complicated than, the usual definition, which corresponds to arbitrarily selecting the pair corresponding to $\beta = 1$. On the other hand it is more general. In particular, the pair $< 1, 0 >$ is well defined whereas with the usual definition it becomes an infinite eigenvalue.

To illustrate the various situations that can arise we consider two by two matrices in the following examples.

Example 9.1 Let

$$A = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

By the definition (α, β) is an eigenvalue if $\det(\beta A - \alpha B) = 0$ which gives the set of pairs (α, β) satisfying the relation $\beta = \pm i\alpha$. In other words, the two generalized eigenvalues are $< 1, i >$ and $< 1, -i >$. This example underscores the fact that the eigenvalues of a symmetric real (or Hermitian complex) pencil are not necessarily real.

Example 9.2 Let

$$A = \begin{pmatrix} -1 & 1 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

Here $\det(\beta A - \alpha B) = \alpha\beta$, so the definition shows that $< 0, 1 >$ and $< 1, 0 >$ are generalized eigenvalues. Note that both matrices are singular.

Example 9.3 Let

$$A = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

In this case any pair $< \alpha, \beta >$ is an eigenvalue since $\det(\beta A - \alpha B) = 0$ independently of the two scalars α and β. Note that this will occur whenever the two matrices are singular and have a common null space. Any vector of the null-space can then be viewed as a degenerate eigenvector associated with an arbitrary scalar. Such pencils are said to be singular.

Example 9.4 Let

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 0 & 2 \\ 0 & 2 \end{pmatrix}.$$

This is another example where any pair (α, β) is an eigenvalue since $\det(\beta A - \alpha B) = 0$ independently of α and β. The two matrices are again singular but here their two null spaces do not intersect. Any 'eigenvalue' (α, β) has the associated 'eigenvector' $(2\alpha, -\beta)^H$.

The above examples suggests an important case that may cause difficulties numerically. This is the case of 'singular pairs'.

Definition 9.1 *A matrix pair (A, B) is called singular if $\beta A - \alpha B$ is singular for all α, β. A matrix pair that is not singular is said to be regular .*

The added complexity due for example to one (or both) of the matrices being singular means that special care must be exercised when dealing with generalized eigen-problems. However, the fact that one or both of the matrices A or B is singular does not mean that trouble is lurking. In fact generalized eigenvalue problem can be quite well behaved in those situations, if handled properly.

We now state a number of definitions and properties. If we multiply both components A and B of the pencil (A, B) to the left by the same nonsingular matrix Y then the eigenvalues and right eigenvectors are preserved. Similarly, if we multiply them to the right by the same non-singular matrix X then the eigenvalues and the left eigenvectors are preserved. The left eigenvectors are multiplied by Y^{-H} in the first case and the right eigenvectors are multiplied by X^{-1} in the second case. These transformations generalize the similarity transformations of the standard problems.

Definition 9.2 *If X and Y are two nonsingular matrices, the pencil (YAX, YBX) is said to be equivalent to the pencil (A, B).*

We will now mention a few properties. Recall that if (α, β) is an eigenvalue pair for (A, B), then $(\bar{\alpha}, \bar{\beta})$ is an eigenvalue pair for (A^H, B^H). The corresponding eigenvector is called the left eigenvector of the pair (A, B).

A rather trivial property, which may have some nontrivial consequences, is that the *eigenvectors* of (A, B) are the same as those of (B, A). An eigenvalue pair (α, β) is simply permuted to (β, α).

In the standard case we know that a left and a right eigenvector associated with two distinct eigenvalues are orthogonal. We will now show a similar property for the generalized problem.

Proposition 9.1 *Let $\lambda_i =< \alpha_i, \beta_i >$ and $\lambda_j =< \alpha_j, \beta_j >$ two distinct generalized eigenvalues of the pair (A, B) and let u_i be a right eigenvector associated with λ_i and w_j a left eigenvector associated with λ_j. Then,*

$$(Au_i, w_j) = (Bu_i, w_j) = 0. \tag{9.5}$$

Proof. Writing the definition for λ_i yields,

$$\beta_i A u_i - \alpha_i B u_i = 0.$$

Therefore,

$$0 = (\beta_i A u_i - \alpha_i B u_i, w_j) = (u_i, (\bar{\beta}_i A^H - \bar{\alpha}_i B^H) w_j) \ . \tag{9.6}$$

We can multiply both sides of the above equation by β_j and use the fact that $(\bar{\alpha}_j, \bar{\beta}_j)$ is an eigenvalue for A^H, B^H with associated eigenvector w_j to get,

$$\begin{aligned} 0 &= (u_i, \bar{\beta}_i \bar{\beta}_j A^H w_j - \bar{\alpha}_i \bar{\beta}_j B^H w_j) \\ 0 &= (u_i, (\bar{\beta}_i \bar{\alpha}_j - \bar{\alpha}_i \bar{\beta}_j) B^H w_j) \\ 0 &= (\beta_i \alpha_j - \alpha_i \beta_j)(B u_i, w_j). \end{aligned}$$

This implies that $(Bu_i, w_j) = 0$ because

$$\beta_i \alpha_j - \alpha_i \beta_j = \begin{vmatrix} \beta_i & \beta_j \\ \alpha_i & \alpha_j \end{vmatrix}$$

must be nonzero by the assumption that the two eigenvalues are distinct. Finally, to show that $(Au_i, w_j) = 0$ we can redo the proof, this time multiplying both sides of (9.6) by α_j instead of β_j, or we can simply observe that we can interchange the roles of A and B, and use the fact that (A, B) and (B, A) have the same set of eigenvectors. ∎

The proposition suggests that when all eigenvalues are distinct, we may be able to simultaneously diagonalize A and B. In fact if all eigenvalues are distinct then the proposition translates into

$$W^H AU = D_A, \quad W^H BU = D_B$$

in which D_A and D_B are two diagonals, U is the matrix of the right eigenvectors and W the matrix of left eigenvectors (corresponding to eigenvalues listed in the same order as for U). There are two points that are still unclear. The first is that we do not know how many distinct eigenvalues there can be. We would like to show that when the pair is regular then there are n of them so that the matrices U and W in the above equality are $n \times n$ matrices. The second point is that we do not know yet whether or not the eigenvectors associated with these distinct eigenvalues are linearly independent. When either A or B are nonsingular then the eigenvectors associated with distinct eigenvectors are linearly independent. This can be seen by observing that the eigenvectors of the pair (A, B) are the same as those of $(B^{-1}A, I)$ in case B is nonsingular or $(I, A^{-1}B)$ when A is nonsingular. As it turns out this extends to the case when the pencil is regular. When the pair (A, B) is a regular pair, then there are two scalars σ_*, τ_* such that the matrix $\tau_* A - \sigma_* B$ is nonsingular. We would like to construct *linearly transformed pairs* that have the same eigenvectors as (A, B) and such that one of the two matrices in the pair is nonsingular. The following theorem will help establish the desired result.

Theorem 9.1 *Let (A, B) any matrix pencil and consider the transformed pencil (A_1, B_1) defined by*

$$A_1 = \tau_1 A - \sigma_1 B \ , \quad B_1 = \tau_2 B - \sigma_2 A \ , \tag{9.7}$$

for any four scalars $\tau_1, \tau_2, \sigma_1, \sigma_2$ such that the 2×2 matrix

$$\Omega = \begin{pmatrix} \tau_2 & \sigma_1 \\ \sigma_2 & \tau_1 \end{pmatrix}$$

is nonsingular. Then the pencil (A_1, B_1) has the same eigenvectors as the pencil (A, B). An associated eigenvalue $(\alpha^{(1)}, \beta^{(1)})$ of the transformed pencil (A_1, B_1) is related to an eigenvalue pair (α, β) of the original pencil (A, B) by

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \Omega \begin{pmatrix} \alpha^{(1)} \\ \beta^{(1)} \end{pmatrix} \ . \tag{9.8}$$

Proof. Writing that $(\alpha^{(1)}, \beta^{(1)})$ is an eigenvalue pair of (A_1, B_1) with associated eigenvector u we get

$$\beta^{(1)}(\tau_1 A - \sigma_1 B)u = \alpha^{(1)}(\tau_2 B - \sigma_2 A)u$$

which after grouping the Au and Bu terms together yields,

$$(\tau_1 \beta^{(1)} + \sigma_2 \alpha^{(1)})Au = (\tau_2 \alpha^{(1)} + \sigma_1 \beta^{(1)})Bu \ . \tag{9.9}$$

The above equation shows that u is an eigenvector for the original pair (A, B) associated with the eigenvalue (α, β) with

$$\beta = \tau_1 \beta^{(1)} + \sigma_2 \alpha^{(1)}, \quad \alpha = \tau_2 \alpha^{(1)} + \sigma_1 \beta^{(1)}. \tag{9.10}$$

Note that (α, β) is related by (9.8) to $(\alpha^{(1)}, \beta^{(1)})$ and as a result α and β cannot both vanish because of the nonsingularity of Ω.

Conversely, to show that any eigenvector of (A, B) is an eigenvector of (A_1, B_1) we can show that A and B can be expressed by relations similar to those in (9.7) in terms of A_1 and B_1. This comes from the fact that Ω is nonsingular.

■

A result of the above theorem is that we can basically identify a regular problem with one for which one of the matrices in the pair is nonsingular. Thus, the choice $\sigma_1 = \sigma_*, \tau_1 = \tau_*$ and $\sigma_2 = \sigma_1, \tau_2 = -\tau_1$ makes the matrix A_1 nonsingular with a non-singular Ω transformation. In fact once τ_1, σ_1 are selected any choice of τ_2 and σ_2 that makes Ω nonsingular will be acceptable.

Another immediate consequence of the theorem is that when (A, B) is regular then there are n eigenvalues (counted with their multiplicities).

Corollary 9.1 *Assume that the pair (A, B) has n distinct eigenvalues. Then the matrices U and W of the n associated right and left eigenvectors respectively, are nonsingular and diagonalize the matrices A and B simultaneously, i.e., there are two diagonal matrices D_A, D_B such that,*

$$W^H A U = D_A \ , \quad W^H B U = D_B \ .$$

The equivalent of the Jordan canonical form is the Weierstrass-Kronecker form. In the following we denote by diag (X, Y) a block diagonal matrix with X in the (1,1) block and Y in the (2,2) block.

Theorem 9.2 *A regular matrix pencil (A, B) is equivalent to a matrix pencil of the form*

$$(\text{diag}\ (J, I)\ ,\ \text{diag}\ (I, N))\ , \tag{9.11}$$

in which the matrices are partitioned in the same manner, and where J and N are in Jordan canonical form and N is nilpotent.

The equivalent of the Schur canonical form would be to simultaneously reduce the two matrices A and B to upper triangular form. This is indeed possible and can be shown by a simple generalization of the proof of Schur's theorem seen in Chapter I.

Theorem 9.3 *For any regular matrix pair* (A, B) *there are two unitary matrices* Q_1 *and* Q_2 *such that*

$$Q_1^H A Q_2 = R_A \quad \textit{and} \quad Q_1^H B Q_2 = R_B$$

are two upper triangular matrices.

2.2. Reduction to Standard Form

When one of the components of the pair (A, B) is nonsingular, there are simple ways to get a standard problem from a generalized one. For example, when B is nonsingular, we can transform the original system

$$\beta A u = \alpha B u$$

into

$$B^{-1} A u = \alpha u$$

taking $\beta = 1$. This simply amounts to multiplying both matrices in the pair by B^{-1}, thus transforming (A, B) into the equivalent pencil $(B^{-1}A, I)$. Other transformations are also possible. For example, we can multiply on the right by B^{-1} transforming (A, B) into the equivalent pair (AB^{-1}, I). This leads to the problem

$$AB^{-1} y = \alpha y \quad \text{with} \quad u = B^{-1} y.$$

Similarly, when A is nonsingular, we can solve the problem

$$A^{-1} B u = \alpha u$$

setting $\beta = 1$ or, again using the variable $y = A^{-1} u$,

$$BA^{-1} y = \alpha y.$$

Note that all the above problems are non Hermitian in general. When A and B are both Hermitian and, in addition, B is positive definite, a better alternative may be to exploit the Choleski factorization of B. If $B = LL^T$, we get after multiplying from the left by L^{-1} and from the right by L^{-T}, the standard problem

$$L^{-1}AL^{-T}y = \alpha y. \tag{9.12}$$

None of the above transformations can be used when both A and B are singular. In this particular situation, one can shift the matrices, i.e., use a transformation of the form described in theorem (9.1). If the pencil is regular then there will be a matrix Ω that will achieve the appropriate transformation. In practice these transformations are not easy to perform since we need to verify whether or not a transformed matrix is singular. If a pencil is regular but both A and B are singular, then chances are that a slight linear transformation will yield a pair with one or both of the matrices nonsingular. However this is not easy to check in practice. First, there is the difficulty of determining whether or not a matrix is deemed nonsingular. Second, in case the two matrices have a nontrivial common null space, then this trial-and-error approach cannot work since any pair α, β will yield a singular $\beta A - \alpha B$, and this information will not be enough to assert that the pair (A, B) is singular.

The particular case where both components A and B are singular and their null spaces have a nontrivial intersection, i.e.,

$$\mathrm{Ker}(A) \cap \mathrm{Ker}(B) \ \neq \ \{0\}$$

deserves a few more words. This is a special singular problem. In practice, it may sometimes be desirable to 'remove' the singularity, and compute the eigenvalues associated with the restriction of the pencil to the complement of the null space. This can be achieved provided we can compute a basis of the common null space, a task that is not an easy one for large sparse matrices, especially if the dimension of the null space is not small.

2.3. Deflation

For practical purposes, it is important to define deflation processes for the generalized eigenvalue problem. In particular we would like to see how we can extend, in the most general setting, the Wielandt deflation procedure seen in Chapter IV. Assuming we have computed an eigenvector u_1 associated with some eigenvalue $\lambda_1 =< \alpha, \beta >$, of (A, B) the most general way of defining analogues of the deflated matrix A_1 of Chapter IV is to deflate the matrices A and B as follows:

$$A_1 = A - \sigma_1 B u_1 v^H , \tag{9.13}$$
$$B_1 = B - \sigma_2 A u_1 v^H . \tag{9.14}$$

We assume, as in the standard case, that $v^H u_1 = 1$. We can easily verify that the eigenvector u_1 is still an eigenvector of the pair (A_1, B_1). The corresponding eigenvalue pair (α', β') must satisfy

$$\beta' A_1 u_1 = \alpha' B_1 u_1$$

from which we get the relation

$$(\beta' + \sigma_2 \alpha') A u_1 = (\alpha' + \sigma_1 \beta') B u_1 .$$

Thus we can identify $\alpha' + \sigma_1 \beta'$ with α and $\beta' + \sigma_2 \alpha'$ with β, to get

$$\alpha = \alpha' + \sigma_1 \beta', \quad \beta = \beta' + \sigma_2 \alpha' . \tag{9.15}$$

Inverting the relations, we get

$$\alpha' = \frac{\alpha - \sigma_1 \beta}{1 - \sigma_1 \sigma_2}, \quad \beta' = \frac{\beta - \sigma_2 \alpha}{1 - \sigma_1 \sigma_2} \tag{9.16}$$

assuming that $1 - \sigma_1 \sigma_2 \neq 0$. The scaling by $1 - \sigma_1 \sigma_2$ can be ignored to obtain the simpler relations,

$$\alpha' = \alpha - \sigma_2 \beta , \quad \beta' = \beta - \sigma_1 \alpha \tag{9.17}$$

which can be rewritten as

$$\begin{pmatrix} \alpha' \\ \beta' \end{pmatrix} = \begin{pmatrix} 1 & -\sigma_1 \\ -\sigma_2 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} . \tag{9.18}$$

In the standard case we have $B = I$, $\beta = \beta' = 1$ and $\sigma_2 = 0$, so the standard eigenvalue is changed to $\alpha' = \alpha - \sigma_1$ as was seen in Chapter IV.

Using Proposition 9.1, we can show that the left eigenvectors not associated with λ_1 are preserved. The particular choice $v = Bw_1$, in which w_1 is the left eigenvector associated with the eigenvalue λ_1 preserves both left and right eigenvectors and is a generalization of Hotelling's deflation, see Exercise P-9.3.

2.4. Shift-and-Invert

Before defining the analogue of the standard shift-and-invert technique we need to know how to incorporate linear shifts. From Theorem 9.1 seen in Section 2.1, for any pair of scalars σ_1, σ_2, the pair $(A - \sigma_1 B, B - \sigma_2 A)$ has the same eigenvectors as the original pair (A, B). An eigenvalue (α', β') of the transformed matrix pair is related to an eigenvalue pair (α, β) of the original matrix pair by

$$\begin{aligned} \alpha &= \alpha' + \sigma_1 \beta' , \\ \beta &= \beta' + \sigma_2 \alpha' . \end{aligned}$$

Computing (α', β') from (α, β) we get, assuming $1 - \sigma_1\sigma_2 \neq 0$,

$$\alpha' = \frac{\alpha - \sigma_1 \beta}{1 - \sigma_1 \sigma_2}, \quad \beta' = \frac{\beta - \sigma_2 \alpha}{1 - \sigma_1 \sigma_2}.$$

In fact, since the eigenvalues are defined up to a scaling factor, we can write

$$\alpha' = \alpha - \sigma_1 \beta , \quad \beta' = \beta - \sigma_2 \alpha . \tag{9.19}$$

It is common to take one of the two shifts, typically σ_2, to be zero. In this special situation:

$$\alpha' = \alpha - \sigma_1\beta \ , \quad \beta' = \beta$$

which gives the usual situation corresponding to $\beta = 1$.

Shift-and-invert for the generalized problems corresponds to multiplying through the two matrices of the shifted pair by the inverse of one of them, typically the first. Thus the shifted-and-inverted pair would be

$$\Big(I \ , \ (A - \sigma_1 B)^{-1}(B - \sigma_2 A) \ \Big) \, .$$

This is now a problem which has the same eigenvalues as the pair $(A - \sigma_1 B, B - \sigma_2 A)$, i.e., its generic eigenvalue pair (α', β') is related to the original pair (α, β) of (A, B) via (9.19). It seems as if we have not gained anything as compared with the pair $(A - \sigma_1 B, B - \sigma_2 A)$. However, the A -matrix for the new pair is the identity matrix.

The most common choice is $\sigma_2 = 0$ and $\sigma_1\beta$ close to an eigenvalue of the original matrix.

2.5. Projection Methods

The projection methods seen in Chapter IV are easy to extend to generalized eigen-problems. In the general framework of oblique projection methods, we are given two subspaces K and L and seek an approximate eigenvector $\tilde{u}$ in the subspace K and an approximate eigenvalue $(\tilde{\alpha}, \tilde{\beta})$ such that

$$(\tilde{\beta} A - \tilde{\alpha} B)\tilde{u} \perp L. \tag{9.20}$$

Given two bases $V = \{v_1, \ldots, v_m\}$, and $W = \{w_1, \ldots, w_m\}$ of K and L, respectively, and writing $\tilde{u} = Vy$, the above conditions translate into the generalized eigenvalue problem

$$\tilde{\beta} W^H A V y = \tilde{\alpha} W^H B V y \ .$$

Note that we can get a standard projected problem if we can find a pair W, V that is such that $W^H BV = I$. For orthogonal projection methods ($K = L$), this will be the case in particular when B is Hermitian positive definite, and the system of vectors $\{v_i\}_{i=1,\ldots m}$ is B-orthonormal.

When the original pencil is Hermitian definite, i.e., when A and B are Hermitian positive definite and when B is positive definite, the projected problem will also be Hermitian definite. The approximate eigenvalues will also be real and all of the properties seen for the Hermitian case in Chapter I will extend in a straight-forward way.

2.6. The Hermitian Definite Case

We devote this section to the important particular case where both A and B are Hermitian and one of them, say B, is positive definite. This situation corresponds to the usual Hermitian eigenproblem in the standard case. For example the eigenvalues are real and the eigenvectors from an orthogonal set with respect to the B–inner product defined by

$$(x, y)_B = (Bx, y) \ . \tag{9.21}$$

That this represents a proper inner product is well-known. The corresponding norm termed the B-norm is given by

$$\|x\|_B = (Bx, x)^{1/2} \ .$$

An important observation that is key to understanding this case is that even though the matrix $C = B^{-1}A$ of one of the equivalent standard eigenproblems is non-Hermitian with respect to the Euclidean inner product, it is *self-adjoint with respect to the B-inner product* in that

$$(Cx, y)_B = (x, Cy)_B \quad \forall \ x, y \ . \tag{9.22}$$

Therefore, one can expect that all the results seen for the standard problem for Hermitian case will be valid provided we replace

Euclidean product by the B-inner product. For example, the min-max theorems will be valid provided we replace the Rayleigh quotient $(Ax,x)/(x,x)$ by

$$\mu(x) = \frac{(Cx,x)_B}{(x,x)_B} = \frac{(Ax,x)}{(Bx,x)} .$$

If we were to use the Lanczos algorithm we would have two options. The first is to factor the B matrix and use the equivalent standard formulation (9.12). This requires factoring the B-matrix and then solving a lower and an upper triangular system at each step of the Lanczos algorithm. An interesting alternative would be to simply employ the standard Lanczos algorithm for the matrix $C = B^{-1}A$ replacing the usual Euclidean inner product by the B inner product at each time that an inner product is invoked. Because of the self-adjointness of C with respect to the B inner product, we will obtain an algorithm similar to the one in the standard case, which is based on a simple three term recurrence. A naive implementation of the main loop in exact arithmetic would consist of the following steps,

$$\begin{aligned} w &:= B^{-1}Av_j , && (9.23)\\ \alpha_j &:= (w, v_j)_B , && (9.24)\\ w &:= w - \alpha_j v_j - \beta_j v_{j-1} , && (9.25)\\ \beta_{j+1} &:= \|w\|_B , && (9.26)\\ v_{j+1} &:= w/\beta_{j+1} . \end{aligned}$$

We observe that α_j in (9.24) is also equal to (Av_j, v_j) and this gives an easy way of computing the $\alpha' s$, using standard Euclidean inner products. Before multiplying Av_j by B^{-1} in (9.23) α_j is computed and saved. The computation on β_{j+1} is a little more troublesome. The use of the definition of the B-inner product would require a multiplication by the matrix B. This may be perfectly acceptable if B is diagonal but could be wasteful in other cases. One way to avoid this matrix product is to observe that, by construction, the vector w in (9.26) is B-orthogonal to the vectors v_j and v_{j-1}. Therefore,

$$(Bw, w) = (Av_j, w) - \alpha_j (Bv_j, w) - \beta_j (Bv_{j-1}, w) = (Av_j, w).$$

As a result, if we save the vector Av_j computed in (9.23) until the end of the loop we can evaluate the B-norm of w with just an Euclidean inner product. Another alternative is to keep a three-term recurrence for the vectors $z_j = Bv_j$. Then Bw is available as

$$Bw = Av_j - \alpha_j z_j - \beta_j z_{j-1}$$

and the inner product (Bw, w) can be evaluated. Normalizing Bw by β_{j+1} yields z_{j+1}. This route requires two additional vectors of storage and a little additional computation but is likely to be more viable from the numerical point of view. Whichever approach is taken, a first algorithm will look as follows.

ALGORITHM **9.1 First Lanczos algorithm for matrix pairs**

1. **Start:** *Choose an initial vector* v_1 *of B-Norm unity. Set* $\beta_1 = 0$, $v_0 = 0$.
2. **Iterate:** *For* $j = 1, 2, \ldots, m$, *do:*
   (a) $v := Av_j$,
   (b) $\alpha_j := (v, v_j)$,
   (c) $w := B^{-1}v - \alpha_j v_j - \beta_j v_{j-1}$,
   (d) *Compute* $\beta_{j+1} = \|w\|_B$, *using* $\beta_{j+1} := \sqrt{(v, w)}$,
   (e) $v_{j+1} = w/\beta_{j+1}$.

One difficulty in the above algorithm is the possible occurrence of a negative B norm of w in the presence of rounding errors.

A second algorithm which is based on keeping a three-term recurrence for the z_j's, implements a modified Gram-Schmidt version of the Lanczos algorithm, i.e., it is analogous to Algorithm 6.5 seen in Chapter VI.

ALGORITHM **9.2 Second Lanczos algorithm for matrix pairs**

1. **Start:** *Choose an initial vector* v_1 *of* B*-Norm unity. Set* $\beta_1 = 0$, $z_0 = v_0 = 0$, $z_1 = Bv_1$.
2. **Iterate:** *For* $j = 1, 2, \ldots, m$, *do*
   (a) $v := Av_j - \beta_j z_{j-1}$,
   (b) $\alpha_j = (v, v_j)$,
   (c) $v := v - \alpha_j z_j$,
   (d) $w := B^{-1}v$,
   (e) $\beta_{j+1} = \sqrt{(w, v)}$,
   (f) $v_{j+1} = w/\beta_{j+1}$ *and* $z_{j+1} = v/\beta_{j+1}$.

Note that the B-norm in (d) is now of the form $(B^{-1}v, v)$ and since B is Hermitian positive definite, this should not cause any numerical problems if computed properly.

In practice the above two algorithms will be unusable in the common situation when B is singular. This situation has been studied carefully in [109]. Without going into the geometric details, we would like to stress that the main idea here is to shift the problem so as to make $(A - \sigma B)$ nonsingular and then work in the subspace $\text{Ran}(A - \sigma B)^{-1}B$. A simplification of the algorithm in [109] is given next. Here, σ is the shift.

ALGORITHM **9.3 Spectral Transformation Lanczos**

1. **Start:** *Choose an initial vector* w *in* $\text{Ran}[\ (A - \sigma B)^{-1}B\ ]$. *Compute* $z_1 = Bw$ *and* $\beta_1 := \sqrt{(w, z_1)}$. *Set* $v_0 := 0$.
2. **Iterate:** *For* $j = 1, 2, \ldots, m$, *do*
   (a) $v_j = w/\beta_j$ and $z_j := z_j/\beta_j$,
   (b) $z_j = (A - \sigma B)^{-1}w$,

(c) $w := w - \beta_j v_{j-1}$,

(d) $\alpha_j = (w, z_j)$,

(e) $w := w - \alpha_j z_j$,

(f) $z_{j+1} = Bw$,

(g) $\beta_{j+1} = \sqrt{(z_{j+1}, w)}$.

Note that the algorithm requires only multiplications with the matrix B. As in the previous algorithm, the two most recent z_j's must be saved, possibly all of them if some form of B - reorthogonalization is to be included. We should point out a simple connection between this algorithm and the previous one. With the exception of the precaution taken to choose the initial vector, algorithm 9.3 is a slight reformulation of Algorithm 9.2, applied to the pair (A', B') where $A' = B$ and $B' = (A - \sigma B)$.

3. Quadratic Problems

The equation of motion for a structural system with viscous damping and without external forces is governed by the equation

$$M\ddot{q} + C\dot{q} + Kq = 0 .$$

In vibration analysis, the generic solution of this equation is assumed to take the form $q = ue^{\lambda t}$ and this leads to the quadratic eigenvalue problem

$$(\lambda^2 M + \lambda C + K)u = 0 . \tag{9.27}$$

These eigenvalue problems arise in dynamical systems where damping and other effects, e.g., gyroscopic, are taken into account. Such effects will define the C matrix. In the next subsections we will see how to adapt some of the basic tools to solve quadratic problems.