Acta Numerica (2002), pp. 519–584
DOI: 10.1017/S0962492902000089

Numerical methods for large eigenvalue problems

Danny C. Sorensen
Department of Computational and Applied Mathematics,
Rice University,
6100 Main St., MS134,
Houston, TX 77005-1892, USA
E-mail: `sorensen@rice.edu`

Over the past decade considerable progress has been made towards the numerical solution of large-scale eigenvalue problems, particularly for nonsymmetric matrices. Krylov methods and variants of subspace iteration have been improved to the point that problems of the order of several million variables can be solved. The methods and software that have led to these advances are surveyed.

CONTENTS

1. Introduction

The algebraic eigenvalue problem

$$\mathbf{Ax} = \mathbf{x}\lambda$$

is fundamental to scientific computing. Large-scale problems are of increasing importance, and recent advances in the area of nonsymmetric problems have enormously expanded capabilities in areas such as linear stability and bifurcation analysis. Considerable progress has been made over the past decade towards the numerical solution of large-scale nonsymmetric problems. However, there is still a great deal to be done. This is a very challenging area of research that is still very active.

This survey is an attempt to introduce some of these advances. It emphasizes two main approaches: Krylov subspace projection and a variant of subspace iteration. Within these two classes, the implicitly restarted Arnoldi method (Sorensen 1992) and the Jacobi–Davidson method (Sleijpen and van der Vorst 1995) are featured. There are several important competing methods but these are discussed in far less detail. Availability of reliable software for large symmetric and nonsymmetric problems has enabled many significant advances in applications. Problems of the order of several million variables are now being solved on massively parallel machines. Problems of order ten thousand can now be solved on a laptop computer. Software and performance issues are therefore a third component of this survey.

Large eigenvalue problems arise in a variety of settings. Two important areas are vibrational analysis of structures and linear stability analysis of fluid flow. The former analysis usually leads to symmetric eigenproblems where the goal typically is to determine the lowest modes. The latter analysis leads to nonsymmetric eigenproblems and the interest is in determining if the eigenvalues lie in a particular half of the complex plane. In both of these settings the discrete problem can become extremely large, but only a few eigenvalues are needed to answer the question of interest.

A typical source of large-scale problems is the discretization of a partial differential equation, for example,

$$\begin{aligned} \mathcal{L}u &= u\lambda \ \text{ for } \ u \in \Omega, \\ u &= 0 \ \text{ for } \ u \in \partial\Omega, \end{aligned} \tag{1.1}$$

where $\mathcal{L}$ is some linear differential operator. Often, $\mathcal{L}$ is a linearization of a nonlinear operator about a particular solution to the nonlinear equation, such as a steady state. A number of techniques may be used to discretize $\mathcal{L}$. The finite element method provides an elegant discretization, and an oversimplified sketch of this discretization follows. If $\mathcal{W}$ is a linear space (or vector space) of functions in which the solution to (1.1) may be found, and $\mathcal{W}_n \subset \mathcal{W}$ is an n-dimensional subspace with basis functions $\{\phi_j\}$, then an approximate solution u_n can be expanded in the form

$$u_n = \sum_{j=1}^{n} \phi_j \xi_j .$$

A variational or Galerkin principle is used, depending on whether $\mathcal{L}$ is self-adjoint, to obtain

$$\left\langle \phi_i, \mathcal{L}\left(\sum_{j=1}^{n} \phi_j \xi_j \right) \right\rangle = \left\langle \phi_i, \sum_{j=1}^{n} \phi_j \xi_j \right\rangle \lambda,$$

where $\langle \cdot, \cdot \rangle$ is an inner product on $\mathcal{W}_n$. This leads to the following systems of equations:

$$\sum_{j=1}^{n} \langle \phi_i, \mathcal{L}\phi_j \rangle \xi_j = \sum_{j=1}^{n} \langle \phi_i, \phi_j \rangle \xi_j \lambda, \tag{1.2}$$

for $1 \le i \le n$. We may rewrite (1.2) and obtain the matrix equation

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}\lambda,$$

where

$$\begin{aligned} \mathbf{A}_{i,j} &= \langle \phi_i, \mathcal{L}\phi_j \rangle, \\ \mathbf{B}_{i,j} &= \langle \phi_i, \phi_j \rangle, \\ \mathbf{x}^T &= [\xi_1, \ldots, \xi_n]^T, \end{aligned}$$

for $1 \le i, j \le n$.

There are several attractive features of a FEM discretization. The boundary conditions are naturally and systematically imposed in a consistent way in the discrete problem. Other important physical properties can also be incorporated into the finite element spaces. Rayleigh quotients with respect to $(\mathbf{A}, \mathbf{B})$ give Rayleigh quotients for $\mathcal{L}$:

$$\frac{\mathbf{v}^*\mathbf{A}\mathbf{v}}{\mathbf{v}^*\mathbf{B}\mathbf{v}} = \frac{\langle \phi, \mathcal{L}\phi \rangle}{\langle \phi, \phi \rangle},$$

where $\phi \in \mathcal{W}_n$ is the function defined by the components of $\mathbf{v}$ as expansion coefficients. Since $\phi \in \mathcal{W}_n \subset \mathcal{W}$, in the self-adjoint case the smallest generalized eigenvalue of $(\mathbf{A}, \mathbf{B})$ is an upper bound for the smallest eigenvalue of the continuous operator $\mathcal{L}$. Typically the basis functions are chosen so that $\mathbf{A}$ and $\mathbf{B}$ are sparse matrices, that is, only a few of the entries in a typical row are nonzero.

In particular, methods for solving the eigenproblem that avoid matrix factorizations and similarity transformations are of interest. The methods discussed here only require matrix-vector products, or perhaps a single sparse direct matrix factorization. Typically, only a few eigenpairs are sought and these methods only require storage proportional to $n \cdot k$, where k is the number of eigenpairs desired. Advantages of such methods are obvious and we list a few:

- sparsity of the matrices is exploited,
- matrices need not be stored – we only need a subroutine for computing the necessary matrix-vector product,
- parallelism is easy.

2. Notation and background

Before discussing methods, we give a brief review to fix notation and introduce basic ideas. We shall consider $n \times n$ square matrices $\mathbf{A}$ with complex entries. The notation $\mathbf{v}^*, \mathbf{A}^*$ will denote the complex conjugate-transpose of a vector (if complex), or the transpose (if real), and likewise for matrices. We shall use $\|\mathbf{v}\|$ to denote the Euclidean norm of a vector $\mathbf{v}$ and $\|\mathbf{A}\|$ to denote the induced matrix two-norm. The real and complex number fields will be denoted by $\mathbb{R}$ and $\mathbb{C}$ respectively. The set of numbers $\sigma(\mathbf{A}) := \{\lambda \in \mathbb{C} : \text{rank}(\lambda\mathbf{I} - \mathbf{A}) < n)\}$ is called the *spectrum* of $\mathbf{A}$. The elements of $\sigma(\mathbf{A})$ are the *eigenvalues* of $\mathbf{A}$ and are the n roots of the *characteristic polynomial* $p_A(\lambda) := \det(\lambda\mathbf{I} - \mathbf{A})$. To each distinct eigenvalue $\lambda \in \sigma(\mathbf{A})$ corresponds at least one nonzero right eigenvector $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{x}\lambda$. A nonzero vector $\mathbf{y}$ such that $\mathbf{y}^*\mathbf{A} = \lambda\mathbf{y}^*$ is called a *left eigenvector.* The *algebraic* multiplicity $n_a(\lambda)$ is the multiplicity of λ as a root of p_A, and the dimension $n_g(\lambda)$ of $\text{Null}(\lambda\mathbf{I} - \mathbf{A})$ is the *geometric* multiplicity of λ. A matrix is *defective* if $n_g(\lambda) < n_a(\lambda)$, for some λ, and otherwise $\mathbf{A}$ is *nondefective.* The eigenvalue λ is *simple* if $n_a(\lambda) = 1$, and $\mathbf{A}$ is *derogatory* if $n_g(\lambda) > 1$ for some λ.

A subspace $\mathcal{S}$ of $\mathbb{C}^{n\times n}$ is an *invariant subspace* of $\mathbf{A}$ if $\mathbf{A}\mathcal{S} \subset \mathcal{S}$. It is straightforward to show that if $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{V} \in \mathbb{C}^{n\times k}$ and $\mathbf{H} \in \mathbb{C}^{k\times k}$ satisfy

$$\mathbf{AV} = \mathbf{VH}, \tag{2.1}$$

then $\mathcal{S} := \text{Range}(\mathbf{V})$ is an invariant subspace of $\mathbf{A}$. Moreover, if $\mathbf{V}$ has full column rank k, then the columns of $\mathbf{V}$ form a basis for this subspace and $\sigma(\mathbf{H}) \subset \sigma(\mathbf{A})$. If $k = n$ then $\sigma(\mathbf{H}) = \sigma(\mathbf{A})$, and $\mathbf{A}$ is said to be *similar* to $\mathbf{H}$. The matrix $\mathbf{A}$ is *diagonalizable* if it is similar to a diagonal matrix. We use the notation $\mathcal{S} = \mathcal{S}_1 \oplus \mathcal{S}_2$ to denote that $\mathcal{S}$ is a *direct sum* of subspaces $\mathcal{S}_1$ and $\mathcal{S}_2$ ($\mathcal{S} = \mathcal{S}_1 + \mathcal{S}_2$ and $\mathcal{S}_1 \cap \mathcal{S}_2 = \{0\}$).

The *Schur decomposition* is fundamental to this discussion and is relevant to some very successful numerical algorithms.

Theorem 2.1. Every square matrix $\mathbf{A}$ possesses a Schur decomposition

$$\mathbf{AQ} = \mathbf{QR}, \tag{2.2}$$

where $\mathbf{Q}$ is unitary ($\mathbf{Q}^*\mathbf{Q} = \mathbf{I}$) and $\mathbf{R}$ is upper triangular. The diagonal elements of $\mathbf{R}$ are the eigenvalues of $\mathbf{A}$.

Schur decompositions are not unique: the eigenvalues of $\mathbf{A}$ may appear on the diagonal of $\mathbf{R}$ in any specified order. From the Schur decomposition, it is easily seen that:

- the matrix $\mathbf{A}$ is normal ($\mathbf{AA}^* = \mathbf{A}^*\mathbf{A}$) if and only if $\mathbf{A} = \mathbf{Q\Lambda Q}^*$ with $\mathbf{Q}$ unitary, and $\mathbf{\Lambda}$ diagonal,
- the matrix $\mathbf{A}$ is Hermitian ($\mathbf{A} = \mathbf{A}^*$) if and only if $\mathbf{A} = \mathbf{Q\Lambda Q}^*$ with $\mathbf{Q}$ unitary, and $\mathbf{\Lambda}$ is diagonal with real diagonal elements.

In either case the eigenvectors of $\mathbf{A}$ are the orthonormal columns of $\mathbf{Q}$ and the eigenvalues are the diagonal elements of $\mathbf{\Lambda}$.

If $\mathbf{V}_k$ represents the leading k columns of $\mathbf{Q}$, and $\mathbf{R}_k$ the leading principal $k \times k$ submatrix of $\mathbf{R}$, then

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{R}_k.$$

This is called a *partial Schur decomposition* of $\mathbf{A}$, and there is always a partial Schur decomposition of $\mathbf{A}$ with the diagonal elements of $\mathbf{R}_k$ consisting of any specified subset of k eigenvalues of $\mathbf{A}$. Moreover, Range$\{\mathbf{V}_k\}$ is an invariant subspace of $\mathbf{A}$ corresponding to these eigenvalues.

3. Single-vector iterations

Single-vector iterations are the simplest and most storage-efficient ways to compute a single eigenvalue and its corresponding eigenvector. The classic power method is the simplest of these and underlies the behaviour of virtually all methods for large-scale problems. This stems from the fact that one is generally restricted to repeated application of a fixed operator to produce a sequence of vectors. The power method is shown in Algorithm 1.

```
Given a nonzero $\mathbf{v}$;
for $k = 1, 2, 3, \dots,$ until convergence
    $\mathbf{w} = \mathbf{A}\mathbf{v}$
    $j = \text{i\_max}(\mathbf{w})$
    $\lambda = \mathbf{w}(j)$
    $\mathbf{v} \leftarrow \mathbf{w}/\lambda$
end
```

Algorithm 1. The power method

This method is suggested by the observation

$$\mathbf{A}^k\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j \lambda_j^k \gamma_j,$$

where $\mathbf{A}\mathbf{q}_j = \mathbf{q}_j\lambda_j$ and $\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j\gamma_j$, and this leads to a straightforward convergence analysis when $\mathbf{A}$ is diagonalizable.

If the eigenvalues of $\mathbf{A}$ are indexed such that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$, then we have

$$\frac{1}{\lambda_1^k}\mathbf{A}^k\mathbf{v}_1 = \mathbf{q}_1\gamma_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k \mathbf{z}_k, \tag{3.1}$$

where $\mathbf{z}_k := \sum_{j=2}^{n} \mathbf{q}_j (\frac{\lambda_j}{\lambda_2})^k \gamma_j$. The ordering of λ_j implies that $\|\mathbf{z}_k\|$ is uniformly bounded. Of course, λ_1 is not available, but it is easily seen that, after k iterations, the contents of $\mathbf{v}$ are

$$\begin{aligned} \mathbf{v} &= \frac{\mathbf{A}^k \mathbf{v}_1}{\mathbf{e}_{j_o}^T \mathbf{A}^k \mathbf{v}_1} \\ &= \frac{\lambda_1^{-k} \mathbf{A}^k \mathbf{v}_1}{\lambda_1^{-k} \mathbf{e}_{j_o}^T \mathbf{A}^k \mathbf{v}_1} \\ &= \frac{\mathbf{q}_1 \gamma_1 + (\frac{\lambda_2}{\lambda_1})^k \mathbf{z}_k}{\mathbf{e}_{j_o}^T (\mathbf{q}_1 \gamma_1 + (\frac{\lambda_2}{\lambda_1})^k \mathbf{z}_k)} \\ &= \mathbf{q}_1 + \mathcal{O}\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \to \mathbf{q}_1, \qquad \text{as} \quad k \to \infty, \end{aligned}$$

where $j_o = \text{i\_max}(\mathbf{q}_1)$ and we assume $\mathbf{q}_1(j_o) = 1$. The function $\text{i\_max}(\mathbf{w})$ selects the index of the first element of largest magnitude. For sufficiently large k, the selection of $j = \text{i\_max}(\mathbf{w})$ in Algorithm 1 returns $j = j_o$ (except in one annoying case that is of no real consequence).

This simple analysis must be modified when $\mathbf{A}$ is defective. In this case the behaviour of powers of Jordan blocks of spectral radius less than one replace the powers of ratios of eigenvalues.

Scaling by the component of largest magnitude facilitates the convergence analysis. We could just as easily scale to the unit ball in any of the standard vector norms. The directions of the vectors $\mathbf{v}$ are the same regardless. Often the eigenvalue estimate is taken to be the Rayleigh quotient $\lambda = \mathbf{v}^* \mathbf{A} \mathbf{v}$, where $\mathbf{v} = \mathbf{w}/\|\mathbf{w}\|$, and this is certainly recommended when $\mathbf{A}$ is Hermitian, since the eigenvalue estimates converge about twice as fast with this estimate.

The two major drawbacks to the power method are the rate of convergence, which is proportional to $|\frac{\lambda_2}{\lambda_1}|$ and can be arbitrarily slow, and that only one eigenvector can be computed.

The problem of slow convergence and convergence to interior eigenvalues may, of course, be remedied by replacing $\mathbf{A}$ by $(\mathbf{A} - \sigma \mathbf{I})^{-1}$, where σ is near an eigenvalue of interest. Later, more will be said about such spectral transformations. To address the problem of obtaining several eigenvectors, deflation schemes have been devised to find a subsequent eigenvector once the first one has converged (Saad 1992). Wielandt deflation is one of these. However, this scheme is not suitable for the nonsymmetric problem.

It is clear that various linear combinations of power iterates might be devised to approximate additional eigenvectors. For example, $\widehat{\mathbf{v}}_j = (\mathbf{v}_j - \mathbf{v}_{j-1} \lambda_1)/\lambda_2$ will converge to a multiple of $\mathbf{q}_2$. However, there is a systematic way to consider all such possibilities at once and pick the optimal one automatically.

4. Krylov subspace projection methods

A systematic way to approach this question is to consider all possible linear combinations of the leading k vectors in the power sequence and ask how the best possible approximate eigeninformation might be extracted. The successive vectors produced by a power iteration may contain considerable information along eigenvector directions corresponding to eigenvalues near the one with largest magnitude. A single-vector power iteration simply ignores this information. Subspace projection provides a way to extract this additional information. Rather than discard the vectors produced during the power iteration, additional eigen-information is obtained by looking at various linear combinations of the power sequence. This immediately leads to consideration of the *Krylov subspace*

$$\mathcal{K}_k(\mathbf{A},\mathbf{v}) := \mathrm{Span}\{\mathbf{v}, \mathbf{A}\mathbf{v}, \mathbf{A}^2\mathbf{v}, \dots, \mathbf{A}^{k-1}\mathbf{v}\},$$

and to seek the best approximate eigenvector that can be constructed from this subspace.

Approximate eigenpairs are constructed by imposing a Galerkin condition. Given any k-dimensional subspace $\mathcal{S}$ of $\mathbb{C}^n$, we define a vector $\mathbf{x} \in \mathcal{S}$ to be a *Ritz vector*, with corresponding *Ritz value* θ, if the Galerkin condition

$$\langle \mathbf{w}, \mathbf{A}\mathbf{x} - \mathbf{x}\theta \rangle = 0, \quad \text{for all} \quad \mathbf{w} \in \mathcal{S}, \tag{4.1}$$

is satisfied, with $\langle \cdot,\cdot \rangle$ denoting some inner product on $\mathbb{C}^n$. In this setting, we are interested in $\mathcal{S} = \mathcal{K}_k(\mathbf{A},\mathbf{v}_1)$. More general subspaces will be considered later.

The definition of $\mathcal{K}_k := \mathcal{K}_k(\mathbf{A},\mathbf{v})$ implies that every $\mathbf{w} \in \mathcal{K}_k$ is of the form $\mathbf{w} = \phi(\mathbf{A})\mathbf{v}_1$ for some polynomial ϕ of degree less than k and also that $\mathcal{K}_{j-1} \subset \mathcal{K}_j$ for $j = 2, 3, \dots, k$. If a sequence of orthogonal bases $\mathbf{V}_j = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_j]$ has been constructed with $\mathcal{K}_j = \mathrm{Range}(\mathbf{V}_j)$ and $\mathbf{V}_j^*\mathbf{V}_j = \mathbf{I}_j$, then it is fairly straightforward to see that $\mathbf{v}_j = \phi_{j-1}(\mathbf{A})\mathbf{v}_1$ where ϕ_{j-1} is a polynomial of degree $j-1$. To extend the basis for $\mathcal{K}_k$ to one for $\mathcal{K}_{k+1}$, a new vector must be constructed with a component in the direction of $\mathbf{A}^k\mathbf{v}_1$ and then orthogonalized with respect to the previous basis vectors. Since $\mathbf{v}_k$ is the only basis vector available with a component in the direction of $\mathbf{A}^{k-1}\mathbf{v}_1$, the new basis vector $\mathbf{v}_{k+1}$ is obtained by

$$\mathbf{f}_k = \mathbf{A}\mathbf{v}_k - \mathbf{V}_k\mathbf{h}_k, \tag{4.2}$$

$$\mathbf{v}_{k+1} = \mathbf{f}_k/\|\mathbf{f}_k\|, \tag{4.3}$$

where the vector $\mathbf{h}_k$ is constructed to achieve $\mathbf{V}_k^*\mathbf{f}_k = 0$. Of course, the orthogonality of the columns of $\mathbf{V}_k$ gives the formula $\mathbf{h}_k = \mathbf{V}_k^*\mathbf{A}\mathbf{v}_k$.

This construction provides a crucial fact concerning $\mathbf{f}_k$:

$$\|\mathbf{f}_k\| = \min_{\mathbf{h}} \|\mathbf{A}\mathbf{v}_k - \mathbf{V}_k\mathbf{h}\| = \min \|\phi(\mathbf{A})\mathbf{v}_1\|, \tag{4.4}$$

where the second minimization is over all polynomials ϕ of degree k with the same leading coefficient as ϕ_{k-1} (*i.e.*, $\lim_{\tau\to\infty} \frac{\tau\phi_{k-1}(\tau)}{\phi(\tau)} = 1$, where $\mathbf{v}_k = \phi_{k-1}(\mathbf{A})\mathbf{v}_1$).

This construction fails when $\mathbf{f}_k = 0$, but in this case

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k,$$

where $\mathbf{H}_k = \mathbf{V}_k^*\mathbf{A}\mathbf{V}_k = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k]$ (with a slight abuse of notation). Hence, this 'good breakdown' happens precisely when $\mathcal{K}_k$ is an invariant subspace of $\mathbf{A}$. The precise conditions that cause $\mathbf{f}_k = 0$ are introduced later in connection with restarting.

4.1. The Arnoldi factorization

The construction leading to the formulas in (4.2) results in the fundamental Arnoldi method for constructing an orthonormal basis for $\mathcal{K}_k$. It expresses a relation between the matrix $\mathbf{A}$, the basis matrix $\mathbf{V}_k$ and the residual vector $\mathbf{f}_k$ of the form

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k + \mathbf{f}_k\mathbf{e}_k^*,$$

where $\mathbf{V}_k \in \mathbb{C}^{n\times k}$ has orthonormal columns, $\mathbf{V}_k^*\mathbf{f}_k = 0$ and $\mathbf{H}_k = \mathbf{V}_k^*\mathbf{A}\mathbf{V}_k$ is a $k \times k$ upper Hessenberg matrix with nonnegative subdiagonal elements. This will be called a *k-step Arnoldi factorization* of $\mathbf{A}$. When $\mathbf{A}$ is Hermitian this implies $\mathbf{H}_k$ is real, symmetric and tridiagonal and then the relation is called a *k-step Lanczos factorization* of $\mathbf{A}$. The columns of $\mathbf{V}_k$ are referred to as the *Arnoldi vectors* or *Lanczos vectors*, respectively.

Ritz pairs satisfying the Galerkin condition (4.1) are derived from the eigenpairs of the small projected matrix $\mathbf{H}_k$. If $\mathbf{H}_k\mathbf{y} = \mathbf{y}\theta$, then the vector $\mathbf{x} = \mathbf{V}_k\mathbf{y}$ satisfies

$$\|\mathbf{A}\mathbf{x} - \mathbf{x}\theta\| = \|(\mathbf{A}\mathbf{V}_k - \mathbf{V}_k\mathbf{H}_k)\mathbf{y}\| = |\beta_k\mathbf{e}_k^*\mathbf{y}|,$$

where $\beta_k := \|\mathbf{f}_k\|$. Observe that if $(\mathbf{x}, \theta)$ is a Ritz pair then

$$\theta = \mathbf{y}^*\mathbf{H}_k\mathbf{y} = (\mathbf{V}_k\mathbf{y})^*\mathbf{A}(\mathbf{V}_k\mathbf{y}) = \mathbf{x}^*\mathbf{A}\mathbf{x}$$

is a Rayleigh quotient (assuming $\|\mathbf{y}\| = 1$), and the associated Rayleigh quotient residual $\mathbf{r}(\mathbf{x}) := \mathbf{A}\mathbf{x} - \mathbf{x}\theta$ satisfies

$$\|\mathbf{r}(\mathbf{x})\| = |\beta_k\mathbf{e}_k^*\mathbf{y}|.$$

When $\mathbf{A}$ is Hermitian, this relation may be used to provide computable rigorous bounds on the accuracy of the eigenvalues of $\mathbf{H}_k$ as approximations to eigenvalues of $\mathbf{A}$ (Parlett 1980). Of course, when $\mathbf{A}$ is non-Hermitian, a small residual does not necessarily imply an accurate approximate eigenpair. Nonnormality effects may corrupt the accuracy. In any case, in exact arithmetic, when $\mathbf{f}_k = 0$ these Ritz pairs become exact eigenpairs of $\mathbf{A}$.

The explicit steps needed to form a k-step Arnoldi factorization are given in Algorithm 2. The factorization is represented visually in Figure 1.

```
$\mathbf{v}_1 = \mathbf{v}/\|\mathbf{v}\|;$
$\mathbf{w} = \mathbf{A}\mathbf{v}_1;\ \ \alpha_1 = \mathbf{v}_1^*\mathbf{w};$
$\mathbf{f}_1 \leftarrow \mathbf{w} - \mathbf{v}_1\alpha_1;$
$\mathbf{V}_1 \leftarrow [\mathbf{v}_1];\ \mathbf{H}_1 \leftarrow [\alpha_1];$
for $j = 1, 2, 3, \ldots, k-1,$
    $\beta_j = \|\mathbf{f}_j\|;\ \mathbf{v}_{j+1} \leftarrow \mathbf{f}_j/\beta_j;$
    $\mathbf{V}_{j+1} \leftarrow [\mathbf{V}_j, \mathbf{v}_{j+1}];$
    $\hat{\mathbf{H}}_j \leftarrow \begin{bmatrix} \mathbf{H}_j \\ \beta_j \mathbf{e}_j^* \end{bmatrix};$
    $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_{j+1};$
    $\mathbf{h} \leftarrow \mathbf{V}_{j+1}^*\mathbf{w};$
    $\mathbf{f}_{j+1} \leftarrow \mathbf{w} - \mathbf{V}_{j+1}\mathbf{h};$
    $\mathbf{H}_{j+1} \leftarrow [\hat{\mathbf{H}}_j, \mathbf{h}];$
end
```

Algorithm 2. k-step Arnoldi factorization

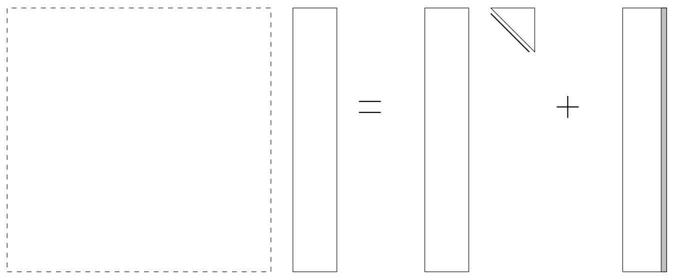

Figure 1. Arnoldi visualization

The formulas given here are based on the *classical Gram–Schmidt* (CGS) orthogonalization process. Often, the orthogonalization is expressed in terms of the *modified Gram–Schmidt* (MGS) process. When the Arnoldi factorization is used to approximate the solution of a linear system, MGS is usually adequate. However, for eigenvalue calculations, the orthogonal basis is very important numerically. In finite precision, MGS does not provide an orthogonal basis and the orthogonality deteriorates in proportion to the condition number of the matrix $[\mathbf{v}, \mathbf{A}\mathbf{v}, \ldots, \mathbf{A}^{k-1}\mathbf{v}]$. In the restarting schemes we shall devise, it is a goal to reach a state of dependence in order to obtain $\mathbf{f}_k = 0$, and MGS is inappropriate for this situation. A second drawback for MGS is that it must be expressed in terms of Level 1 BLAS (Lawson, Hanson, Kincaid and Krogh 1979).

When expressed in terms of CGS, the dense matrix-vector products $\mathbf{V}_{j+1}^*\mathbf{w}$ and $\mathbf{V}_{j+1}\mathbf{h}$ may be coded in terms of the Level 2 BLAS operation _GEMV (Dongarra, DuCroz, Hammarling and Hanson 1988). This provides a significant performance advantage on virtually every platform from workstation to supercomputer.

Unfortunately, the CGS process is notoriously unstable and will fail miserably in this setting without modification. However, it can be rescued via a technique proposed by Daniel, Gragg, Kaufman and Stewart (DGKS) in 1976. This provides an excellent way to construct a vector $\mathbf{f}_{j+1}$ that is numerically orthogonal to $\mathbf{V}_{j+1}$. It amounts to computing a correction

$$\mathbf{c} = \mathbf{V}_{j+1}^*\mathbf{f}_{j+1}; \quad \mathbf{f}_{j+1} \leftarrow \mathbf{f}_{j+1} - \mathbf{V}_{j+1}\mathbf{c}; \quad \mathbf{h} \leftarrow \mathbf{h} + \mathbf{c};$$

just after the construction of $\mathbf{f}_{j+1}$ if necessary. One may perform a simple test to avoid this DGKS correction if it is not needed. The correction only needs to be computed if $\|\mathbf{h}\| < \eta(\|\mathbf{h}\|^2 + \|\mathbf{f}_{j+1}\|^2)^{1/2}$, where $0 < \eta < 1$ is a specified parameter. The test ensures that the new vector $\mathbf{A}\mathbf{v}$ makes an angle greater than $\cos^{-1}\eta$ with the existing Krylov subspace. This mechanism maintains orthogonality to full working precision at very reasonable cost. The special situation imposed by the restarting scheme we are about to discuss makes this modification essential for obtaining accurate eigenvalues and numerically orthogonal Schur vectors (eigenvectors in the Hermitian case). This scheme is visualized in Figure 2, where it is shown that the initial projection $\mathbf{V}\mathbf{h}$ of $\mathbf{w} = \mathbf{A}\mathbf{v}$ is the exact projection of a perturbed vector. The correction vector $\mathbf{c}$ then corrects the non-orthogonal vector $\mathbf{f} = \mathbf{w} - \mathbf{V}\mathbf{h}$ to a new one $\mathbf{f}_+ \leftarrow \mathbf{f} - \mathbf{V}\mathbf{c}$ that is orthogonal to Range($\mathbf{V}$).

It has been well documented that failure to maintain orthogonality leads to numerical difficulties. In the Hermitian case, Paige (1971) showed that the loss of orthogonality occurs precisely when an eigenvalue of $\mathbf{H}_j$ is close to an eigenvalue of $\mathbf{A}$. In fact, the Lanczos vectors lose orthogonality in the direction of the associated approximate eigenvector. Failure to maintain orthogonality results in spurious copies of the approximate eigenvalue

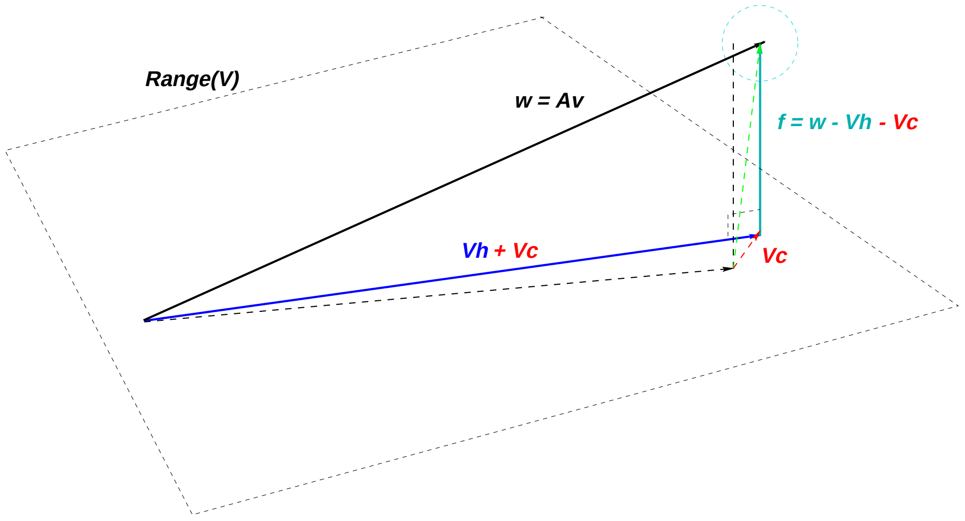

Figure 2. DGKS correction

produced by the Lanczos method (Algorithm 4). Implementations based on selective and partial orthogonalization (Grimes, Lewis and Simon 1994, Parlett and Scott 1979, Simon 1984) monitor the loss of orthogonality and perform additional orthogonalization steps only when necessary. The methods developed in Cullum and Willoughby (1981, 1985) and in Parlett and Reid (1981) use the three-term recurrence with no re-orthogonalization steps. Once a level of accuracy has been achieved, the spurious copies of computed eigenvalues are located and deleted. Then the Lanczos basis vectors are regenerated from the three-term recurrence and Ritz vectors are recursively constructed in place. This is a very competitive strategy when the matrix-vector product $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}$ is relatively inexpensive.

4.2. Restarting the Arnoldi process

A clear difficulty with the Lanczos/Arnoldi process is that the number of steps required to calculate eigenvalues of interest within a specified accuracy cannot be predetermined. This depends completely on the starting vector $\mathbf{v}_1$, and generally eigen-information of interest does not appear until k gets very large. In Figure 3 the distribution in the complex plane of the Ritz values (shown in grey dots) is compared with the spectrum (shown as +s). The original matrix is a normally distributed random matrix of order 200 and the Ritz values are from a ($k = 50$)-step Arnoldi factorization. Note that hardly any Ritz values appear in the interior and also that very few eigenvalues of $\mathbf{A}$ are well approximated. Eigenvalues at the extremes of the spectrum of $\mathbf{A}$ are clearly better approximated than the others.

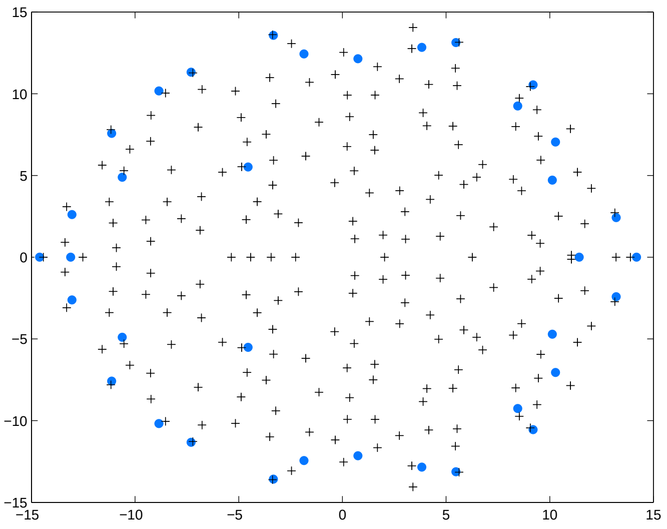

Figure 3. Typical distribution of Ritz values

For large problems, it is clearly intractable to compute and store a numerically orthogonal basis set $\mathbf{V}_k$ for large k. Storage requirements are $\mathcal{O}(n \cdot k)$ and arithmetic costs are $\mathcal{O}(n \cdot k^2)$ flops to compute the basis vectors plus $\mathcal{O}(k^3)$ flops to compute the eigensystem of $\mathbf{H}_k$.

To control this cost, restarting schemes have been developed that iteratively replace the starting vector $\mathbf{v}_1$ with an 'improved' starting vector $\mathbf{v}_1^+$, and then compute a new Arnoldi factorization of fixed length k. Beyond the obvious motivation to control computational cost and storage overheads, there is a clear interest in forcing $\mathbf{f}_k = 0$. However, this is useful only if the spectrum $\sigma(\mathbf{H}_k)$ has the desired properties. The structure of $\mathbf{f}_k$ guides the strategy. The goal is to iteratively force $\mathbf{v}_1$ to be a linear combination of eigenvectors of interest.

Since $\mathbf{v}_1$ determines the subspace $\mathcal{K}_k$, this vector must be constructed to select the eigenvalues of interest. The following lemmas serve as a guide.

Lemma 4.1. If $\mathbf{v} = \sum_{j=1}^{k} \mathbf{q}_j \gamma_j$ where $\mathbf{A}\mathbf{q}_j = \mathbf{q}_j \lambda_j$, and

$$\mathbf{AV} = \mathbf{VH} + \mathbf{f}\mathbf{e}_k^T$$

is a k-step Arnoldi factorization with unreduced $\mathbf{H}$, then $\mathbf{f} = 0$ and $\sigma(\mathbf{H}) = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$.

This lemma follows easily from the observation that $\phi(\mathbf{A})\mathbf{v}_1 = 0$ with $\phi(\tau) = \prod_{i=1}^{k}(\tau - \lambda_j)$ together with the minimization property (4.4), which implies $\mathbf{f}_k = 0$. (An upper Hessenberg matrix $\mathbf{H}$ is *unreduced* if no element of the first subdiagonal is zero.) A more precise statement is as follows.

Lemma 4.2. $\mathbf{f}_k = 0$ if and only if $\mathbf{v}_1 = \mathbf{Q}_k\mathbf{y}$, where $\mathbf{AQ}_k = \mathbf{Q}_k\mathbf{R}_k$ is a partial Schur decomposition of $\mathbf{A}$ with $\mathbf{R}_k$ non-derogatory. Moreover, the Ritz values of $\mathbf{A}$ with respect to $\mathcal{K}_k$ are eigenvalues of $\mathbf{A}$, and are given by the diagonal elements of $\mathbf{R}_k$.

Thus, a more general and superior numerical strategy is to force the starting vector to be a linear combination of Schur vectors that span the desired invariant subspace.

Restarting was initially proposed by Karush (1951) soon after the Lanczos algorithm appeared (Lanczos 1950). Subsequently, there were developments by Paige (1971) Cullum and Donath (1974) and Golub and Underwood (1977) Then, Saad (1984) developed a polynomial restarting scheme for eigenvalue computation based on the acceleration scheme of Manteuffel (1978) for the iterative solution of linear systems.

4.3. Polynomial restarting

Polynomial restarting strategies replace $\mathbf{v}_1$ by

$$\mathbf{v}_1 \leftarrow \psi(\mathbf{A})\mathbf{v}_1,$$

where ψ is a polynomial constructed to damp unwanted components from the starting vector. If $\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j\gamma_j$ where $\mathbf{Aq}_j = \mathbf{q}_j\lambda_j$, then

$$\mathbf{v}_1^+ = \psi(\mathbf{A})\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j\gamma_j\psi(\lambda_j).$$

The idea is to force the starting vector to be ever closer to an invariant subspace, by constructing ψ so that $\psi(\lambda)$ is as small as possible on a region containing the unwanted eigenvalues. This is motivated by Lemmas 4.1 and 4.2. Because of this effect of filtering out (damping) the unwanted components, we often refer to these polynomials as *filter polynomials*, and we refer to their roots as *filter shifts*. The reason for this terminology will become clear when we introduce implicit restarting.

An iteration is defined by repeatedly restarting until the updated Arnoldi factorization eventually contains the desired eigenspace. For more information on the selection of effective restarting vectors, see Saad (1992). One of the more successful approaches is to use Chebyshev polynomials in order to damp unwanted eigenvector components in the available subspace.

Explicit restarting techniques are easily parallelized, in contrast to the overheads involved in implicit restarting (Section 4.4). The reason is that a major part of the work is in matrix-vector products. When we have to solve the eigenproblem on a massively parallel computer for a matrix that allows inexpensive matrix-vector products, this may be an attractive property.

Two possibilities for constructing ψ suggest themselves immediately. One is to construct the polynomial to be 'small' in magnitude on the unwanted set of eigenvalues and large on the wanted set. This criterion can be met by constructing a polynomial that best approximates 0 on a specified set that encloses the unwanted set and excludes the wanted set of eigenvalues. The other possibility is to use the best available approximation to the wanted eigenvectors. These are the Ritz vectors, and so it makes sense to select the current Ritz vectors corresponding to Ritz values that best approximate the wanted eigenvalues, and form

$$\mathbf{v}_+ = \sum_{j=1}^{k} \hat{\mathbf{q}}_j \gamma_j. \tag{4.5}$$

Since each Ritz vector is of the form $\hat{\mathbf{q}}_j = \phi_j(\mathbf{A})\mathbf{v}$, where ϕ_j is a polynomial of degree $j-1 < m$, this mechanism is also a polynomial restart. In Saad (1992), heuristics are given for choosing the weights γ_j.

A third way is to specify the polynomial ψ by its roots. A fairly obvious choice is to find the eigenvalues θ_j of the projected matrix $\mathbf{H}$ and sort these into two sets according to a given criterion: the wanted set $\mathcal{W} = \{\theta_j : j = 1, 2, \ldots, k\}$ and the unwanted set $\mathcal{U} = \{\theta_j : j = k+1, k+2, \ldots, k+p\}$. Then we specify the polynomial ψ as the polynomial with these unwanted Ritz values as its roots. This choice of roots, called *exact shifts*, was suggested in Sorensen (1992).

Morgan (1996) found a remarkable property of this strategy. If exact shifts are used to define $\psi(\tau) = \prod_{j=k+1}^{k+p}(\tau - \theta_j)$, then the Krylov space generated by $\mathbf{v}_1^+ = \psi(\mathbf{A})\mathbf{v}_1$ satisfies

$$\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1^+) = \mathrm{Span}\{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_k, \mathbf{A}\hat{\mathbf{q}}_j, \mathbf{A}^2\hat{\mathbf{q}}_j, \ldots, \mathbf{A}^p\hat{\mathbf{q}}_j\},$$

for any $j = 1, 2, \ldots, k$. Thus polynomial restarting with exact shifts will generate a new subspace that contains all of the possible choices in (4.5).

This property follows from the fact that $\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1^+) = \psi(\mathbf{A})\mathcal{K}_m(\mathbf{A}, \mathbf{v}_1)$, together with the fact that a Ritz vector $\hat{\mathbf{q}}_j$ has the form

$$\hat{\mathbf{q}}_j = \prod_{\substack{i=1 \\ i\neq j}}^{k} (\mathbf{A} - \theta_i \mathbf{I})\psi(\mathbf{A})\mathbf{v}_1,$$

and thus

$$\mathbf{A}^\ell \hat{\mathbf{q}}_j = \mathbf{A}^\ell \prod_{\substack{i=1 \\ i\neq j}}^{k} (\mathbf{A} - \theta_i \mathbf{I})\mathbf{v}_1^+ \in \mathcal{K}_m(\mathbf{A}, \mathbf{v}_1^+), \quad \text{for} \quad \ell = 1, 2, \ldots, p.$$

Hence

$$\mathrm{Span}\{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_k, \mathbf{A}\hat{\mathbf{q}}_j, \mathbf{A}^2\hat{\mathbf{q}}_j, \ldots, \mathbf{A}^p\hat{\mathbf{q}}_j\} \subset \mathcal{K}_m(\mathbf{A}, \mathbf{v}_1^+).$$

A minimal polynomial argument may then be used to establish the linear independence of $\{\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \ldots, \hat{\mathbf{q}}_k, \mathbf{A}\hat{\mathbf{q}}_j, \mathbf{A}^2\hat{\mathbf{q}}_j, \ldots, \mathbf{A}^p\hat{\mathbf{q}}_j\}$, and thus a dimension argument establishes the desired equality. When wanted Ritz values are not distinct, generalized eigenvectors enter into this discussion.

Exact shifts have proved to perform remarkably well in practice and have been adopted as the shift selection of choice when no other information is available. However, there are many other possibilities. For example, if we knew of a region containing the wanted eigenvalues, we might be able to construct filter shifts designed to assure that the filter polynomial would ultimately be very small (in absolute value) over that region. If the containment region were a line segment or an ellipse, we could construct the Chebyshev points related to that region. Another distribution of filter shifts that can be designed for very general containment regions are the Leja points. These have been studied extensively in the literature and have been applied very successfully in the context of an implicitly restarted Lanczos method (IRLM) by Baglama, Calvetti and Reichel (1996). These points figure prominently in the convergence analysis we give in Section 5.

4.4. Implicit restarting

A straightforward way to implement polynomial restarting is to explicitly construct the starting vector $\mathbf{v}_1^+ = \psi(\mathbf{A})\mathbf{v}_1$ by applying $\psi(\mathbf{A})$ through a sequence of matrix-vector products. However, there is an alternative implementation that provides a more efficient and numerically stable formulation. This approach, called *implicit restarting*, uses a sequence of implicitly shifted QR steps to an m-step Arnoldi or Lanczos factorization to obtain a truncated form of the implicitly shifted QR-iteration. Numerical difficulties and storage problems normally associated with Arnoldi and Lanczos processes are avoided. The algorithm is capable of computing a small number k of eigenvalues with user-specified features such as largest real part or largest magnitude using $2nk + \mathcal{O}(k^2)$ storage. The computed Schur basis vectors for the desired k-dimensional eigenspace are numerically orthogonal to working precision.

Implicit restarting enables the extraction of desired eigenvalues and vectors from high-dimensional Krylov subspaces while avoiding the standard storage and numerical difficulties. Desired eigen-information is continually compressed into a fixed-size k-dimensional subspace through an implicitly shifted QR mechanism. An Arnoldi factorization of length $m = k + p$,

$$\mathbf{A}\mathbf{V}_m = \mathbf{V}_m\mathbf{H}_m + \mathbf{f}_m\mathbf{e}_m^*, \tag{4.6}$$

is compressed to a factorization of length k that retains the eigen-information of interest.

QR steps are used to apply p linear polynomial factors $\mathbf{A} - \mu_j\mathbf{I}$ implicitly to the starting vector $\mathbf{v}_1$. The first stage of this shift process results in

$$\mathbf{A}\mathbf{V}_m^+ = \mathbf{V}_m^+\mathbf{H}_m^+ + \mathbf{f}_m\mathbf{e}_m^*\mathbf{Q}, \tag{4.7}$$

where $\mathbf{V}_m^+ = \mathbf{V}_m\mathbf{Q}$, $\mathbf{H}_m^+ = \mathbf{Q}^*\mathbf{H}_m\mathbf{Q}$, and $\mathbf{Q} = \mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_p$. Each $\mathbf{Q}_j$ is the orthogonal matrix associated with implicit application of the shift $\mu_j = \theta_{k+j}$. Since each of the matrices $\mathbf{Q}_j$ is Hessenberg, it turns out that the first $k-1$ entries of the vector $\mathbf{e}_m^*\mathbf{Q}$ are zero (*i.e.*, $\mathbf{e}_m^*\mathbf{Q} = [\sigma\mathbf{e}_k^T, \hat{\mathbf{q}}^*]$). Hence, the leading k columns in equation (4.7) remain in an Arnoldi relation and provide an updated k-step Arnoldi factorization

$$\mathbf{A}\mathbf{V}_k^+ = \mathbf{V}_k^+\mathbf{H}_k^+ + \mathbf{f}_k^+\mathbf{e}_k^*, \tag{4.8}$$

with an updated residual of the form $\mathbf{f}_k^+ = \mathbf{V}_m^+\mathbf{e}_{k+1}\hat{\beta}_k + \mathbf{f}_m\sigma$. Using this as a starting point, it is possible to apply p additional steps of the Arnoldi process to return to the original m-step form.

Virtually any explicit polynomial restarting scheme can be applied with implicit restarting, but considerable success has been obtained with exact shifts. Exact shifts result in $\mathbf{H}_k^+$ having the k wanted Ritz values as its spectrum. As convergence takes place, the subdiagonals of $\mathbf{H}_k$ tend to zero and the most desired eigenvalue approximations appear as eigenvalues of the leading $k \times k$ block of $\mathbf{R}$ as a partial Schur decomposition of $\mathbf{A}$. The basis vectors $\mathbf{V}_k$ tend to numerically orthogonal Schur vectors.

The basic IRAM iteration is shown in Algorithm 3. The expansion and contraction process of the IRAM iteration is visualized in Figure 4.

4.5. Convergence of IRAM

There is a fairly straightforward intuitive explanation of how this repeated updating of the starting vector $\mathbf{v}_1$ through implicit restarting might lead to convergence. If $\mathbf{v}_1$ is expressed as a linear combination of eigenvectors $\{\mathbf{q}_j\}$ of $\mathbf{A}$, then

$$\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j\gamma_j \Rightarrow \psi(\mathbf{A})\mathbf{v}_1 = \sum_{j=1}^{n} \mathbf{q}_j\psi(\lambda_j)\gamma_j.$$

Applying the same polynomial (*i.e.*, using the same shifts) repeatedly for ℓ iterations will result in the jth original expansion coefficient being attenuated by a factor

$$\left(\frac{\psi(\lambda_j)}{\psi(\lambda_1)}\right)^\ell,$$

where the eigenvalues have been ordered according to decreasing values of $|\psi(\lambda_j)|$. The leading k eigenvalues become dominant in this expansion and the remaining eigenvalues become less and less significant as the iteration proceeds. Hence, the starting vector $\mathbf{v}_1$ is forced into an invariant subspace as desired. The adaptive choice of ψ provided with the exact shift mechanism further enhances the isolation of the wanted components in this expansion. Hence, the wanted eigenvalues are approximated ever better as the iteration

```
Compute an $m = k + p$ step Arnoldi factorization
    $\mathbf{AV}_m = \mathbf{V}_m\mathbf{H}_m + \mathbf{f}_m\mathbf{e}_m^*$.
**repeat until** convergence,
    Compute $\sigma(\mathbf{H}_m)$ and select $p$
    shifts $\mu_1, \mu_2, \ldots, \mu_p$;
    $\mathbf{Q} = \mathbf{I}_m$;
    **for** $j = 1, 2, \ldots, p$,
        Factor $[\mathbf{Q}_j, \mathbf{R}_j] = \mathrm{qr}(\mathbf{H}_m - \mu_j\mathbf{I})$;
        $\mathbf{H}_m \leftarrow \mathbf{Q}_j^*\mathbf{H}_m\mathbf{Q}_j$;
        $\mathbf{Q} \leftarrow \mathbf{QQ}_j$;
    **end**
    $\hat{\beta}_k = \mathbf{H}_m(k+1, k)$;  $\sigma_k = \mathbf{Q}(m, k)$;
    $\mathbf{f}_k \leftarrow \mathbf{v}_{k+1}\hat{\beta}_k + \mathbf{f}_m\sigma_k$;
    $\mathbf{V}_k \leftarrow \mathbf{V}_m\mathbf{Q}(:, 1\colon k)$;   $\mathbf{H}_k \leftarrow \mathbf{H}_m(1\colon k, 1\colon k)$;
    Beginning with the $k$-step Arnoldi factorization
        $\mathbf{AV}_k = \mathbf{V}_k\mathbf{H}_k + \mathbf{f}_k\mathbf{e}_k^*$,
    apply $p$ additional steps of the Arnoldi process
    to obtain a new $m$-step Arnoldi factorization
        $\mathbf{AV}_m = \mathbf{V}_m\mathbf{H}_m + \mathbf{f}_m\mathbf{e}_m^*$.
**end**
```

Algorithm 3. Implicitly restarted Arnoldi method (IRAM)

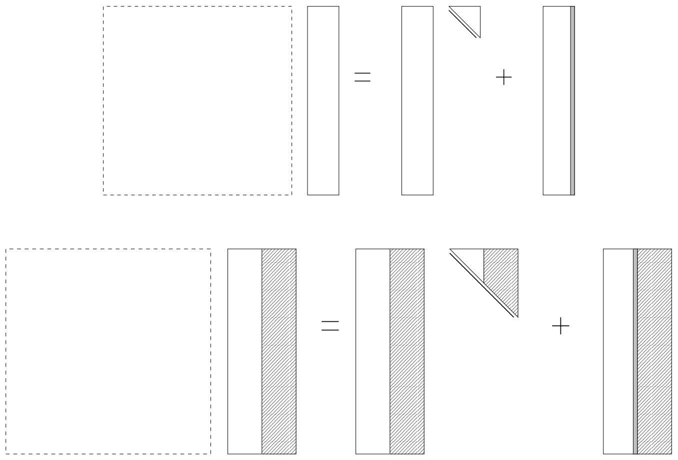

Figure 4. Visualization of IRAM

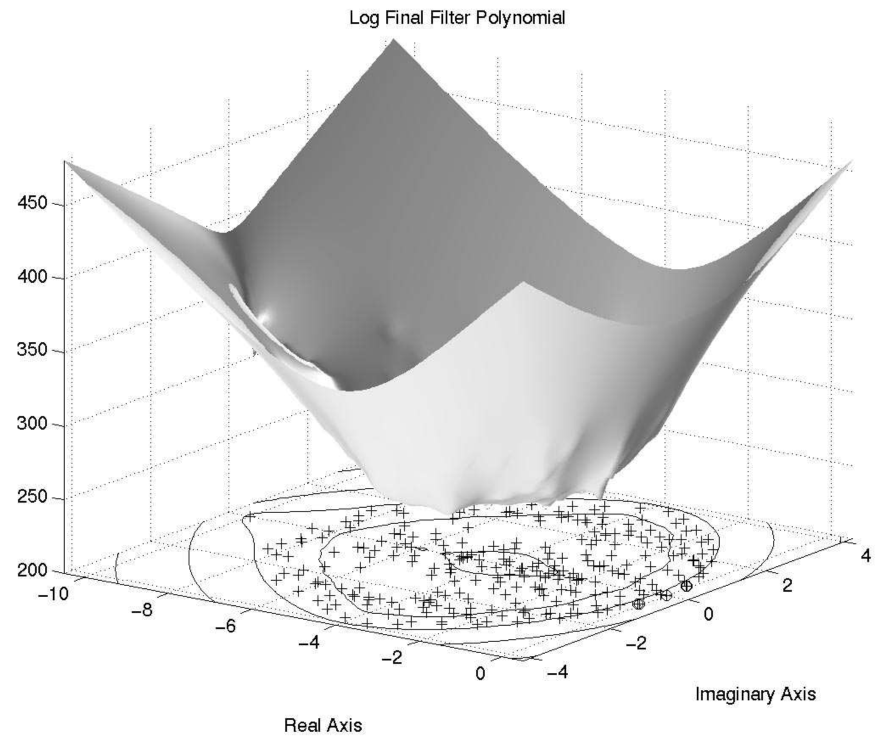

Figure 5. Final filter polynomial from IRAM

proceeds. Unfortunately, making this heuristic argument precise has turned out to be quite difficult. Some fairly sophisticated analysis is required to understand convergence of these methods. In Section 5 we sketch such an analysis for polynomial restarting.

Another way to look at this procedure is to consider the aggregate polynomial

$$\mathbf{v}_{\text{final}} = \Phi(\mathbf{A})\mathbf{v}_1,$$

where $\Phi(\tau)$ is the product of all the polynomials that were applied during the course of the computation. In Figure 5, the plot shows the surface of $\log|\Phi(\tau)|$ for τ in a region of the complex plane containing the eigenvalues of $\mathbf{A}$ (shown by +s). The circled eigenvalues are the five eigenvalues of largest real part that were computed. The filter polynomial Φ was automatically constructed, through the choice of filter shifts, to be small on the unwanted portion of the spectrum and to enhance the wanted portion (the five eigenvalues of largest real part).

We can also learn a great deal by considering a plot of the totality of all the filter shifts in relation to the final converged eigenvalues. This is shown in Figure 6 (a different example than shown in Figure 5). The plot shows all

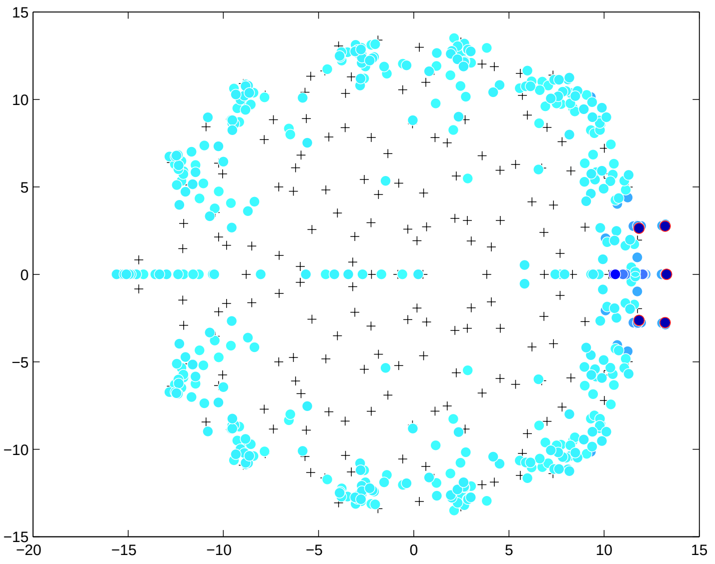

Figure 6. Distribution of filter shifts in IRAM

of the filter shifts (the light dots) applied, and the converged eigenvalues as the five darkest points to the right. The actual eigenvalues of $\mathbf{A}$ are shown as dark $+$s.

It is worth noting that if $m = n$ then $\mathbf{f}_m = 0$, and this iteration is precisely the same as the implicitly shifted QR iteration. Even for $m < n$, the first k columns of $\mathbf{V}_m$ and the Hessenberg submatrix $\mathbf{H}_m(1\colon k, 1\colon k)$ are mathematically equivalent to the matrices that would appear in the full implicitly shifted QR iteration using the same shifts μ_j. In this sense, the implicitly restarted Arnoldi method may be viewed as a truncation of the implicitly shifted QR iteration. The fundamental difference is that the standard implicitly shifted QR iteration selects shifts to drive subdiagonal elements of $\mathbf{H}_n$ to zero from the bottom up, while the shift selection in the implicitly restarted Arnoldi method is made to drive subdiagonal elements of $\mathbf{H}_m$ to zero from the top down.

This implicit scheme costs p rather than the $k+p$ matrix-vector products the explicit scheme would require. Thus the exact shift strategy can be viewed both as a means for damping unwanted components from the starting vector and also for directly forcing the starting vector to be a linear combination of wanted eigenvectors. See Sorensen (1992) for information on the convergence of IRAM and Baglama *et al.* (1996) and Stathopoulos, Saad and Wu (1998) for other possible shift strategies for Hermitian $\mathbf{A}$. The reader is referred to Lehoucq and Scott (1996) and Morgan (1996) for studies comparing implicit restarting with other schemes.

4.6. Deflation schemes for IRAM

The performance of IRAM can be improved considerably with the introduction of appropriate deflation schemes to isolate approximate invariant subspaces associated with converged Ritz values. These deflation strategies can make it possible to compute multiple or clustered eigenvalues with a single-vector implicit restart method.

Since IRAM may be viewed as a truncation of the standard implicitly shifted QR-iteration, it inherits a number of desirable properties. These include some well-understood deflation rules that are extremely important with respect to convergence and stability. These deflation rules are essential for the QR-iteration to efficiently compute multiple or clustered eigenvalues. These rules simply specify a numerically stable criterion for setting small subdiagonal elements of $\mathbf{H}$ to zero. While these existing QR deflation rules are applicable to IRAM, they are not the most effective schemes possible. Here, we introduce additional deflation schemes that are better suited to implicit restarting.

In the large-scale setting, it is highly desirable to provide schemes that can deflate with user-specified relative error tolerances ϵ_D that are perhaps considerably greater than working precision ϵ_M. Without this capability, excessive and unnecessary computational effort is often required to detect and deflate converged approximate eigenvalues. The ability to deflate at relaxed tolerances provides an effective way to compute multiple or clustered eigenvalues with a single-vector implicitly restarted Arnoldi method.

We shall introduce two forms of deflation. The first, a *locking* operation, decouples converged approximate eigenvalues and associated invariant subspaces from the active part of the IRAM iteration. The second, a *purging* operation, removes unwanted but converged eigenpairs. Locking has the effect of isolating an approximate eigenspace once it has converged to a certain level of accuracy and then forcing subsequent Arnoldi vectors to be orthogonal to the converged subspace. With this capability, additional instances of a multiple eigenvalue can be computed to the same specified accuracy without the expense of converging them to unnecessarily high accuracy. Purging allows the deletion of converged but unwanted Ritz values and vectors from the Krylov space when they are not purged naturally by the restarting scheme. With the aid of these deflation schemes, convergence of the IRAM iteration can be greatly improved. Computational effort is also reduced. These notions and appropriate methods were developed in Lehoucq (1995) and Lehoucq and Sorensen (1996). Here, we present a slightly improved variant of those deflation schemes.

Small subdiagonal elements of $\mathbf{H}$ may occur during implicit restarting. However, it is usually the case that there are converged Ritz values appearing in the spectrum of $\mathbf{H}$ long before small subdiagonal elements appear.

This convergence is usually detected through observation of a small last component in an eigenvector $\mathbf{y}$ of $\mathbf{H}$.

It turns out that in the case of a small last component of $\mathbf{y}$, there is an orthogonal similarity transformation of $\mathbf{H}$ that will give an equivalent Arnoldi factorization with a slightly perturbed $\mathbf{H}$ that does indeed have a zero subdiagonal, and this is the basis of our deflation schemes.

Orthogonal deflating transformations

Our deflation schemes rely on the construction of a special orthogonal transformation. As in Lehoucq (1995) and Lehoucq and Sorensen (1996), the deflation is related to an eigenvector $\mathbf{y}$ associated with the Ritz value to be deflated. In the following discussion, it is *very important* to note that the eigenvector $\mathbf{y}$ in either the locking or purging *need not be accurate.* All that is required for successful deflation schemes is that $\|\mathbf{H}\mathbf{y} - \mathbf{y}\theta\| \le \epsilon_M \|\mathbf{H}\|$ in the case of locking, and that $\|\mathbf{y}^*\mathbf{H} - \theta\mathbf{y}^*\| \le \epsilon_M \|\mathbf{H}\|$ in the case of purging, to obtain backward stable deflation rules.

The construction is based on a sequence of Givens transformations. If $\mathbf{y}$ is a given vector of unit length, compute a sequence of plane rotations $\mathbf{G}_{1,j},\ j = 2, \ldots, n$ such that

$$\mathbf{y}_j^* = \mathbf{y}_{j-1}^*\mathbf{G}_{1,j} = (\tau_j, 0, \ldots, 0, \eta_{j+1}, \ldots, \eta_n),$$

beginning with $\mathbf{y}_1 := \mathbf{y}$ and ending with $\mathbf{y}_n = \mathbf{e}_1$. Thus, the orthogonal matrix $\mathbf{Q} = \mathbf{G}_{1,2}\mathbf{G}_{1,3}\cdots\mathbf{G}_{1,n}$ satisfies $\mathbf{y}^*\mathbf{Q} = \mathbf{e}_1^*$. Recalling that each $\mathbf{G}_{1,j}$ is the identity matrix $\mathbf{I}_n$ with the $(1,1), (j,j)$ entries replaced with $\gamma_j,\ \bar{\gamma}_j$ and the $(1,j), (j,1)$ entries replaced with $-\bar{\sigma}_j,\ \sigma_j$, where $|\gamma_j|^2 + |\sigma_j|^2 = 1$, it is easily seen that

$$\mathbf{Q}\mathbf{e}_1 = \mathbf{y} \quad \text{and} \quad \mathbf{e}_k^*\mathbf{Q} = (\eta, \tau\mathbf{e}_{k-1}^*), \tag{4.9}$$

with $|\eta|^2 + |\tau|^2 = 1$.

Locking or purging a single eigenvalue

Now, we shall use the orthogonal transformations developed above to construct stable and efficient transformations needed to implement locking and purging. The simplest case to consider is the treatment of a single eigenvalue. When working in complex arithmetic, this will suffice. Handling complex conjugate eigenvalues of a real nonsymmetric matrix in real arithmetic is a bit more complicated but essentially follows the same theme to deflate two vectors at once.

Locking $\boldsymbol{\theta}$. The first instance to discuss is the locking of a single converged Ritz value. Assume that

$$\mathbf{H}\mathbf{y} = \mathbf{y}\theta, \quad \|\mathbf{y}\| = 1,$$

with $\mathbf{e}_k^*\mathbf{y} = \eta$, where $|\eta| \le \epsilon_D \|\mathbf{H}\|$. Here, it is understood that $\epsilon_M \le \epsilon_D < 1$ is a specified relative accuracy tolerance between ϵ_M and 1.

If θ is 'wanted' then it is desirable to lock θ. However, in order to accomplish this it will be necessary to arrange a transformation of the current Arnoldi factorization to one with a small subdiagonal to isolate θ. This may be accomplished by constructing a $k \times k$ orthogonal matrix $\mathbf{Q} = \mathbf{Q}(\mathbf{y})$ as above with properties (4.9).

Consider the matrix $\mathbf{H}_+ = \mathbf{Q}^*\mathbf{H}\mathbf{Q}$. The first column of this matrix is

$$\mathbf{H}_+\mathbf{e}_1 = \mathbf{Q}^*\mathbf{H}\mathbf{Q}\mathbf{e}_1 = \mathbf{Q}^*\mathbf{H}\mathbf{y} = \mathbf{Q}^*\mathbf{y}\theta = \mathbf{e}_1\theta.$$

Thus $\mathbf{H}_+$ is of the form

$$\mathbf{H}_+ = \begin{bmatrix} \theta & \mathbf{h}^* \\ 0 & \widehat{\mathbf{H}} \end{bmatrix}.$$

We may return the matrix to Hessenberg form using orthogonal similarity transformations without destroying the desirable structure of the last row of $\mathbf{Q}$. One way to accomplish this is to apply a succession of orthogonal transformations of the form

$$\widehat{\mathbf{U}}_j = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathbf{U}_j & 0 \\ 0 & 0 & \mathbf{I}_{k-j} \end{bmatrix},$$

so that $\mathbf{H}_+ \leftarrow \widehat{\mathbf{U}}_j^*\mathbf{H}\widehat{\mathbf{U}}_j$ is constructed to introduce zeros in positions $2, \ldots, j-1$ of row $j+1$ for $j = k-1, k-2, \ldots, 3$. This is a standard Householder reduction to Hessenberg form working from the bottom up. Of course, the orthogonal matrix $\mathbf{Q}$ must be updated in the same way to give $\mathbf{Q} \leftarrow \mathbf{Q}\widehat{\mathbf{U}}_j$, $j = k-1, k-2, \ldots, 3$. On completion, the kth row of $\mathbf{Q}$ remains undisturbed from the original construction.

The end result of these transformations is

$$\mathbf{A}[\mathbf{v}_1, \mathbf{V}_2] = [\mathbf{v}_1, \mathbf{V}_2] \begin{bmatrix} \theta & \mathbf{h}^* \\ 0 & \mathbf{H}_2 \end{bmatrix} + [\mathbf{f}\eta, \mathbf{f}\tau\mathbf{e}_{k-1}^*],$$

where $[\mathbf{v}_1, \mathbf{V}_2] = \mathbf{V}\mathbf{Q}$. Moreover, this relation may be rearranged to give

$$[\mathbf{A} - \mathbf{f}\eta\mathbf{v}_1^*][\mathbf{v}_1, \mathbf{V}_2] = [\mathbf{v}_1, \mathbf{V}_2] \begin{bmatrix} \theta & \mathbf{h}^* \\ 0 & \mathbf{H}_2 \end{bmatrix} + \mathbf{f}\tau\mathbf{e}_k^*,$$

to see that we have an exactly deflated Arnoldi factorization of a nearby matrix $\widehat{\mathbf{A}} = \mathbf{A} - \mathbf{f}\eta\mathbf{v}_1^*$ (remember, η is small).

Now, subsequent implicit restarting steps take place only in the last $k-1$ columns, as if we had

$$\mathbf{A}\mathbf{V}_2 = \mathbf{V}_2\mathbf{H}_2 + \mathbf{f}\tau\mathbf{e}_{k-1}^*,$$

with all the subsequent orthogonal matrices and column deletions associated with implicit restarting applied to $\mathbf{h}^*$, and never disturbing the relation $\mathbf{A}\mathbf{v}_1 = \mathbf{v}_1\theta + \mathbf{f}\eta$. Therefore, if $\widehat{\mathbf{Q}}$ represents a $(k-1) \times (k-1)$ orthogonal matrix associated with an implicit restart, then

$$\mathbf{A}\mathbf{V}_2\widehat{\mathbf{Q}} = (\mathbf{v}_1, \mathbf{V}_2\widehat{\mathbf{Q}}) \begin{pmatrix} \mathbf{h}^*\widehat{\mathbf{Q}} \\ \widehat{\mathbf{Q}}^*\mathbf{H}_2\widehat{\mathbf{Q}} \end{pmatrix} + \mathbf{f}\tau\mathbf{e}_{k-1}^*\widehat{\mathbf{Q}}.$$

In subsequent Arnoldi steps, $\mathbf{v}_1$ participates in the orthogonalization so that the selective orthogonalization recommended by Parlett and Scott (Parlett and Scott 1979, Parlett 1980) is accomplished automatically.

Purging $\boldsymbol{\theta}$. If θ is 'unwanted' then we may wish to remove θ from the spectrum of the projected matrix $\mathbf{H}$. This purging process is required since the implicit restart strategy using exact shifts will sometimes fail to purge a converged unwanted Ritz value (Lehoucq and Sorensen 1996).

The purging process is quite analogous to the locking process just described. However, in this case it is advantageous to use a left eigenvector to obtain the deflation. Let $\mathbf{y}$ be a left eigenvector of $\mathbf{H}$ corresponding to θ, that is,

$$\mathbf{y}^*\mathbf{H} = \theta\mathbf{y}^*.$$

Just as before, we construct a $(k \times k)$ orthogonal matrix $\mathbf{Q}$ such that

$$\mathbf{y}^*\mathbf{Q} = \mathbf{e}_1^*, \quad \text{and} \quad \mathbf{e}_k^*\mathbf{Q} = (\eta, 0, \ldots, 0, \tau),$$

where $\eta = \mathbf{e}_k^*\mathbf{y}$ and $|\tau|^2 + |\eta|^2 = 1$.

Again, consider the matrix $\mathbf{H}_+ = \mathbf{Q}^*\mathbf{H}\mathbf{Q}$. The first row of this matrix is

$$\mathbf{e}_1^*\mathbf{H}_+ = \mathbf{e}_1^*\mathbf{Q}^*\mathbf{H}\mathbf{Q}\mathbf{e}_1 = \mathbf{y}^*\mathbf{H}\mathbf{Q} = \theta\mathbf{y}^*\mathbf{Q} = \theta\mathbf{e}_1^*.$$

Thus $\mathbf{H}_+$ is of the form

$$\mathbf{H}_+ = \begin{bmatrix} \theta & 0 \\ \mathbf{h} & \widehat{\mathbf{H}} \end{bmatrix},$$

and thus

$$\mathbf{A}[\mathbf{v}_k, \widehat{\mathbf{V}}] = [\mathbf{v}_k, \widehat{\mathbf{V}}] \begin{bmatrix} \theta & 0 \\ \mathbf{h} & \widehat{\mathbf{H}} \end{bmatrix} + \mathbf{f}(\eta, \tau\mathbf{e}_{k-1}^*),$$

where $[\mathbf{v}_k, \widehat{\mathbf{V}}] = \mathbf{V}\mathbf{Q}$. Now, simply delete the first column on both sides to get

$$\mathbf{A}\widehat{\mathbf{V}} = \widehat{\mathbf{V}}\widehat{\mathbf{H}} + \mathbf{f}\tau\mathbf{e}_{k-1}^*.$$

We may return this to an Arnoldi factorization as before by constructing an orthogonal $\widehat{\mathbf{Q}}$ with $\mathbf{e}_{k-1}^*\widehat{\mathbf{Q}} = \mathbf{e}_{k-1}^*$ such that $\widehat{\mathbf{Q}}^*\widehat{\mathbf{H}}\widehat{\mathbf{Q}}$ is upper Hessenberg.

In fact, we can use the structure of $\mathbf{Q}$ to show that $\widehat{\mathbf{H}}$, surprisingly, must be upper Hessenberg automatically. However, there are subtleties in achieving an implementation that attains this numerically.

Recently, Stewart has introduced an implicit restarting method that may well resolve the issue of locking and purging. This is presented in Stewart (2001).

4.7. The Lanczos method

As previously mentioned, when $\mathbf{A}$ is Hermitian ($\mathbf{A} = \mathbf{A}^*$) then the projected matrix $\mathbf{H}$ is tridiagonal and the Arnoldi process reduces to the Lanczos method. Historically, the Lanczos process preceded the Arnoldi process.

In the Hermitian case, if we denote the subdiagonal elements of $\mathbf{H}$ by $\beta_1, \beta_2, \ldots, \beta_{n-1}$ and the diagonal elements by $\alpha_1, \alpha_2, \ldots, \alpha_n$, then the relation

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k + \mathbf{f}_k\mathbf{e}_k^*$$

gives

$$\begin{aligned}\mathbf{f}_k &= \mathbf{v}_{k+1}\beta_k \\ &= \mathbf{A}\mathbf{v}_k - \mathbf{v}_k\alpha_k - \mathbf{v}_{k-1}\beta_{k-1}.\end{aligned}$$

This famous three-term recurrence has been studied extensively since its inception. The numerical difficulties are legendary, with the two main issues being the numerical orthogonality of the sequence of basis vectors and the almost certain occurrence of 'spurious' copies of converged eigenvalues reappearing as eigenvalues of the projected matrix $\mathbf{H}_k$.

The most favourable form of the recurrence, in the absence of any additional attempt at achieving orthogonality, is displayed in Algorithm 4. This organization amounts to the last two steps of a modified Gram–Schmidt variant of the Arnoldi process. Mathematically, all of the other coefficients that would ordinarily appear in the Arnoldi process are zero in the Hermitian case and this condition is forced to obtain the Lanczos process.

Once the tridiagonal matrix $\mathbf{H}_m$ has been constructed, analysed and found to possess k converged eigenvalues $\{\theta_1, \theta_2, \ldots, \theta_k\}$, with corresponding eigenvectors $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_k]$, we may construct the eigenvectors with the recursion given in Algorithm 5.

This mechanism is quite attractive when the matrix-vector product $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}$ is relatively inexpensive. However, there are considerable numerical difficulties to overcome. Cullum and Willoughby developed schemes for analysing the projected matrix $\mathbf{H}_m$ and modifying it to get rid of the spurious eigenvalue cases. Briefly, this analysis consists of deleting the first row and column of $\mathbf{H}$ and then comparing the Ritz values of the new $\widehat{\mathbf{H}}$ with those of the original $\mathbf{H}$. Those that are the same are the spurious ones. The heuristic idea is that convergence of the 'good' Ritz values is triggered by significant components in the starting vector. A converged Ritz vector is composed from basis vectors in the Krylov subspace, and these basis vectors only contain (at least in exact arithmetic) components of the converged Ritz vector if the starting vector has a nonzero component in that direction. Since the starting vector is one of the orthogonal basis vectors $\mathbf{V}$ for the Krylov subspace, deleting it from the basis should tend

```
$\mathbf{v}_1 = \mathbf{v}/\|\mathbf{v}\|$;
$\mathbf{w} = \mathbf{A}\mathbf{v}_1$;
$\alpha_1 = \mathbf{v}_1^*\mathbf{w}$;
$\mathbf{f}_1 \leftarrow \mathbf{w} - \mathbf{v}_1\alpha_1$;
for $j = 1, 2, 3, \ldots, m-1$,
    $\beta_j = \|\mathbf{f}_j\|$;
    $\mathbf{v}_{j+1} \leftarrow \mathbf{f}_j/\beta_j$;
    $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_{j+1} - \mathbf{v}_j\beta_j$;
    $\alpha_{j+1} = \mathbf{v}_{j+1}^*\mathbf{w}$;
    $\mathbf{f}_{j+1} \leftarrow \mathbf{w} - \mathbf{v}_{j+1}\alpha_{j+1}$;
end
```

Algorithm 4. The Lanczos process

```
$\mathbf{X} = \mathbf{v}_1\mathbf{Y}(1,:)$;
$\mathbf{w} = \mathbf{A}\mathbf{v}_1$;
$\mathbf{f}_1 \leftarrow \mathbf{w} - \mathbf{v}_1\alpha_1$;
for $j = 1, 2, 3, \ldots, m-1$,
    $\beta_j = \|\mathbf{f}_j\|$;
    $\mathbf{v}_{j+1} \leftarrow \mathbf{f}_j/\beta_j$;
    $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{v}_{j+1}\mathbf{Y}(j+1,:)$;
    $\mathbf{f}_{j+1} \leftarrow \mathbf{A}\mathbf{v}_{j+1} - \mathbf{v}_{j+1}\alpha_{j+1} - \mathbf{v}_j\beta_j$;
end
```

Algorithm 5. Eigenvector recovery in the Lanczos process

to remove important components that triggered convergence. Deleting the first row and column of $\mathbf{H} = \mathbf{V}^*\mathbf{A}\mathbf{V}$ gives an orthogonal projection $\widehat{\mathbf{H}}$ of $\mathbf{A}$ onto a subspace that is orthogonal to the starting vector. Consequently, if a Ritz value persists as an eigenvalue of $\widehat{\mathbf{H}}$, it must be spurious and therefore is the result of rounding errors.

Parlett and Reid (1981) suggested another mechanism to detect convergence of Ritz values, by constructing intervals that must contain an eigenvalue. The advantage of this approach is that it also identifies true eigenvalues that are discovered as a result of rounding errors (for instance when the starting vector was unintentionally orthogonal to the corresponding eigenvector).

Even with this convergence test, there is no assurance of numerical orthogonality of the converged eigenvectors. Parlett and Scott advocate a selected orthogonalization procedure (Parlett and Scott 1979) that orthogonalizes against converged Ritz vectors as they appear. An excellent account of the complete process is given in Parlett (1980). Grimes *et al.* (1994) advocate always using shift-invert, so that the Lanczos sequence is relatively short, and the separation of the transformed eigenvalues aids in the orthogonality, so that a selective orthogonalization scheme is quite successful.

4.8. Harmonic Ritz values and vectors

As we have seen, Ritz values usually approximate the extremal values of the spectrum well, but give poor approximations to interior eigenvalues. One attempt to better approximate interior eigenvalues has been the introduction of *harmonic Ritz values.* These were formally introduced in Paige, Parlett and van der Vorst (1995) for symmetric matrices, but have previously been used for analysis and computation in Morgan (1991) and Freund (1992). In particular, harmonic Ritz values have been proposed for restarting strategies when interior eigenvalues are sought.

There are various ways to introduce this notion. A fairly intuitive idea is to consider Rayleigh quotients of $\mathbf{A}^{-1}$ of the form

$$\theta = \frac{\mathbf{w}^*\mathbf{A}^{-1}\mathbf{w}}{\mathbf{w}^*\mathbf{w}} \quad \text{with} \quad \mathbf{w} \in \mathcal{S},$$

where $\mathcal{S}$ is a well-chosen subspace. A convenient choice is $\mathcal{S} = \mathbf{A}\mathcal{K}_k(\mathbf{A}, \mathbf{v})$. If $\mathbf{w} \in \mathcal{S}$ then $\mathbf{w} = \mathbf{AVy}$ for some $\mathbf{y}$, and

$$\begin{aligned}
\theta &= \frac{\mathbf{w}^*\mathbf{A}^{-1}\mathbf{w}}{\mathbf{w}^*\mathbf{w}} \\
&= \frac{\mathbf{y}^*\mathbf{V}^*\mathbf{A}^*\mathbf{A}^{-1}\mathbf{AVy}}{\mathbf{y}^*\mathbf{V}^*\mathbf{A}^*\mathbf{AVy}} \\
&= \frac{\mathbf{y}^*\mathbf{V}^*\mathbf{A}^*\mathbf{Vy}}{\mathbf{y}^*\mathbf{V}^*\mathbf{A}^*\mathbf{AVy}} \\
&= \frac{\mathbf{y}^*\mathbf{H}^*\mathbf{y}}{\mathbf{y}^*[\mathbf{VH} + \mathbf{f}\mathbf{e}_k^*]^*[\mathbf{VH} + \mathbf{f}\mathbf{e}_k^*]\mathbf{y}} \\
&= \frac{\mathbf{y}^*\mathbf{H}^*\mathbf{y}}{\mathbf{y}^*(\mathbf{H}^*\mathbf{H} + \beta^2\mathbf{e}_k\mathbf{e}_k^*)\mathbf{y}},
\end{aligned}$$

where $\beta = \|\mathbf{f}\|$. Thus θ is a generalized Rayleigh quotient for the matrix pencil $(\mathbf{H}^*, \mathbf{H}^*\mathbf{H} + \beta^2\mathbf{e}_k\mathbf{e}_k^*)$. The harmonic Ritz values are defined as the generalized eigenvalues associated with this pencil. Since θ is related to eigenvalues of $\mathbf{A}^{-1}$, it is natural to define the harmonic Ritz values μ to be the reciprocals of the critical points θ of this Rayleigh quotient. Thus the harmonic Ritz values are the eigenvalues

$$(\mathbf{H}^*\mathbf{H} + \beta^2\mathbf{e}_k\mathbf{e}_k^*)\mathbf{y} = \mathbf{H}^*\mathbf{y}\mu,$$

and the corresponding harmonic Ritz vectors are

$$\mathbf{x} = \mathbf{A}\mathbf{V}\mathbf{y} = \mathbf{V}\mathbf{H}\mathbf{y} + \mathbf{f}(\mathbf{e}_k^*\mathbf{y}).$$

When $\mathbf{H}$ is nonsingular, this simplifies to

$$(\mathbf{H} + \mathbf{g}\mathbf{e}_k^*)\mathbf{y} = \mathbf{y}\mu \quad \text{with} \quad \mathbf{g} = \beta^2\mathbf{H}^{-*}\mathbf{e}_k.$$

When $\mathbf{A}$ is Hermitian and indefinite, with λ_- the largest negative eigenvalue and λ_+ the smallest positive eigenvalue of $\mathbf{A}$, then

$$\mu_- \leq \lambda_- < 0 \quad \text{and} \quad 0 < \lambda_+ \leq \mu_+,$$

with μ_- the largest negative and μ_+ the smallest positive harmonic Ritz values. In other words, the largest interval containing 0 but no eigenvalues of $\mathbf{A}$ is also devoid of any harmonic Ritz values.

The harmonic Ritz values have some interesting properties. For symmetric matrices, the Ritz values converge monotonically to exterior eigenvalues of $\mathbf{A}$. In contrast, the harmonic Ritz values converge monotonically, albeit often at a slow rate, to the interior eigenvalues of $\mathbf{A}$ closest to the origin. This property is intuitively attractive, but has not really resulted in an effective way to compute interior eigenvalues from a Krylov subspace generated by $\mathbf{A}$. We need somehow to generate a subspace that really does contain vectors that can approximate eigenvectors associated with interior eigenvalues. Morgan (1991) has suggested harmonic Ritz vectors for restarts, and the idea has also been incorporated in the Jacobi–Davidson algorithm (Sleijpen and van der Vorst 1996). The latter method does introduce vectors into the subspace that approximate inverse iteration directions, and hence the harmonic Ritz vectors have a better chance of being effective in that setting.

5. Convergence of polynomial restart methods

For nonsymmetric problems, convergence of Krylov projection methods has been studied extensively, but the general case has been elusive. Saad (1980) developed a bound for matrices with simple eigenvalues for the gap between a single eigenvector and the Krylov subspace (gap will be defined below). This result was generalized in Jia (1995) to include defective matrices, but the bounds explicitly involve the Jordan canonical form and derivatives of approximating polynomials. Simoncini (1996) analyses convergence of a block Arnoldi method for defective matrices using pseudospectra. Lehoucq (2001) relates IRAM to subspace iteration to analyse convergence to an invariant subspace. Calvetti, Reichel and Sorensen (1994) introduce concepts from potential theory to analyse IRLM convergence to a single eigenvector for Hermitian matrices.

In the nonsymmetric case, the possibility of nonnormality complicates the analysis considerably. The possibility of derogatory matrices (an eigenvalue with geometric multiplicity greater than one) may even render certain invariant subspaces unreachable. These concepts are introduced in Beattie, Embree and Rossi (2001). They employ various ideas from functional analysis, pseudospectra and potential theory. Their analysis focuses on convergence in gap (a generalized notion of angle) of a (restarted) Krylov space to a desired invariant subspace of $\mathbf{A}$, and they are able to treat convergence in full generality.

In this section, we give a modified version of their results. Our purpose here is to lay out the main ideas as simply as possible while retaining the general theme and content of those excellent results. In doing so, we sacrifice some rigour and our convergence estimates are not as refined. We strongly recommend that the interested reader consult that reference for a thorough and insightful treatment of the convergence issues.

Before launching into this discussion, some motivation is in order. If the matrix $\mathbf{A}$ is normal, then its eigensystem is perfectly conditioned (insensitive to perturbations). In this case, it makes perfect sense to phrase convergence analysis in terms of eigenvalues. However, even in this case, when there are multiple eigenvalues, it makes more sense numerically to phrase such results in terms of convergence to invariant subspaces. In the nonsymmetric case, there is a possibility of a nontrivial Jordan form. If, for example, $\mathbf{A}$ has a Jordan form with a block of order $\ell > 1$, then certain eigenvalues of $\mathbf{A} + \mathbf{E}$ are likely to be perturbed by as much as $\|\mathbf{E}\|^{(1/\ell)}$ from the eigenvalues of $\mathbf{A}$. The best we can hope for in a numerical algorithm is to compute the exact eigensystem of a slightly perturbed matrix of this form with $\|\mathbf{E}\| = \|\mathbf{A}\|\mathcal{O}(\epsilon_M)$ (machine precision). Convergence results based on damping out specific eigenvalues (the unwanted set) in the presence of such perturbations are numerically meaningless. We must, instead, phrase such results in terms of convergence of invariant subspaces and also provide a mechanism to encompass such perturbations. This perturbation theory for nonnormal matrices is perhaps best described in Trefethen (1992, 1999). However, there are several important related papers. For really fascinating computational studies on this topic we heartily recommend the software `MATLAB Pseudospectra GUI, 2000-2001` by T. G. Wright, available at `www.comlab.ox.uk/pseudospectra/pasgui`. Pseudospectra will play a fundamental role in the following discussion.

The following is an attempt to provide a completely general convergence analysis based on the theory presented in Beattie *et al.* (2001). The development here is, admittedly, not entirely rigorous. The intent is to sketch the main ideas in a comprehensive way that can be readily understood.

5.1. Some preliminaries

We are naturally concerned with the Krylov subspace generated by a given starting vector. Note that, for any starting vector $\mathbf{v}_1$, there is a least positive integer k such that

$$\mathcal{K}_k(\mathbf{A}, \mathbf{v}_1) = \mathcal{K}(\mathbf{A}, \mathbf{v}_1) = \mathrm{Span}\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, \ldots\}.$$

Moreover, k is the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\mathbf{v}_1$. This is the monic polynomial ϕ of least degree such that $\phi(\mathbf{A})\mathbf{v}_1 = 0$. From this property, it is straightforward to see that

$$\mathbf{A}\mathbf{K} = \mathbf{K}\mathbf{A}_k, \quad \text{with} \quad \mathbf{K} = [\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \ldots, \mathbf{A}^{k-1}\mathbf{v}_1],$$

where $\mathbf{A}_k = \mathbf{J} + \mathbf{g}\mathbf{e}_k^*$, where $\mathbf{J}$ is a Jordan matrix of order k with ones on the first subdiagonal and zeros elsewhere, and where $\mathbf{g}^T = (\gamma_0, \gamma_1, \ldots, \gamma_{k-1})$ with $\phi(\tau) = \tau^k - \gamma_{k-1}\tau^{k-1} - \cdots - \gamma_1\tau - \gamma_0$. This implies that $\mathcal{K}_k(\mathbf{A}, \mathbf{v}_1)$ is an invariant subspace with respect to $\mathbf{A}$ and that $\mathcal{K}_j(\mathbf{A}, \mathbf{v}_1) \subset \mathcal{K}_k(\mathbf{A}, \mathbf{v}_1)$ for all positive integers j.

Since $\mathbf{A}_k$ is non-derogatory (every eigenvalue of $\mathbf{A}_k$ has geometric multiplicity 1), this observation shows that it is impossible to capture more than a single Jordan block associated with a given eigenvalue. Indeed, if $\mathbf{A}$ is derogatory, then it is technically impossible to compute the entire invariant subspace corresponding to an eigenvalue of geometric multiplicity greater than one, from such a Krylov space. We would necessarily need to employ deflation and restart techniques in this case. In practice, round-off error usually blurs this situation.

To develop an understanding of convergence, a minimal amount of machinery needs to be established. Let λ_j, $1 \leq j \leq N$ be the distinct eigenvalues of $\mathbf{A}$ and let n_j be the algebraic multiplicity of λ_j. From a Schur decomposition $\mathbf{A} = \mathbf{Q}\mathbf{R}\mathbf{Q}^*$ (recall that the eigenvalues λ_j may appear in any specified order on the diagonal of $\mathbf{R}$), we can construct a spectral decomposition

$$\mathbf{A} = \mathbf{X}\widehat{\mathbf{R}}\mathbf{Y}^*, \quad \text{with} \quad \mathbf{Y}^*\mathbf{X} = \mathbf{X}\mathbf{Y}^* = \mathbf{I},$$

where $\widehat{\mathbf{R}}$ is block diagonal with upper triangular blocks $\mathbf{R}_j = \lambda_j\mathbf{I}_{n_j} + \mathbf{U}_j$, and

$$\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N], \quad \mathbf{Y} = [\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N].$$

This construction is detailed in Golub and Van Loan (1996).

The property $\mathbf{Y}^*\mathbf{X} = \mathbf{X}\mathbf{Y}^* = \mathbf{I}$ implies that each $\mathbf{P}_j := \mathbf{X}_j\mathbf{Y}_j^*$ is a projector with the following properties:

$$\begin{aligned}
\mathbf{A}\mathbf{P}_j &= \mathbf{P}_j\mathbf{A} = \mathbf{X}_j\mathbf{R}_j\mathbf{Y}_j^*, \\
\mathbf{A}\mathcal{S}_j &\subset \mathcal{S}_j := \mathrm{Range}(\mathbf{P}_j), \\
\mathbf{I} &= \mathbf{P}_1 + \mathbf{P}_2 + \cdots + \mathbf{P}_N, \\
\mathbb{C}_n &= \oplus_{j=1}^N \mathcal{S}_j.
\end{aligned}$$

With polynomial restart techniques, we attempt to modify the starting vector $\mathbf{v}_1$ in a systematic way to force the invariant subspace $\mathcal{K} := \mathcal{K}_k(\mathbf{A}, \mathbf{v}_1)$ ever closer to a desired invariant subspace $\mathcal{X}_g$ corresponding to wanted eigenvalues $\lambda_j,\ 1 \leq j \leq L$. In keeping with the notation of Beattie *et al.* (2001) we shall denote this selected set as 'good' eigenvalues and the remaining ones will be called 'bad' eigenvalues. It is important to note that there is no assumption about algebraic or geometric multiplicity of these eigenvalues.

Naturally, we are interested in some measure of nearness of $\mathcal{K}$ to $\mathcal{X}_g$. One such device is the *gap* between subspaces. The quantity

$$\delta(\mathcal{W}, \mathcal{V}) := \sup_{\mathbf{w} \in \mathcal{W}} \inf_{\mathbf{v} \in \mathcal{V}} \frac{\|\mathbf{w} - \mathbf{v}\|}{\|\mathbf{w}\|}$$

is called the *containment gap* between subspaces $\mathcal{W}$ and $\mathcal{V}$. We can show that $\delta(\mathcal{W}, \mathcal{V}) = \sin(\theta)$, where θ is the largest canonical angle between a closest subspace $\hat{\mathcal{V}}$ of $\mathcal{V}$ having the same dimension as $\mathcal{W}$. If $\mathcal{W}$ and $\mathcal{V}$ are both one-dimensional, then θ is the angle between unit vectors $\mathbf{v} \in \mathcal{V}$ and $\mathbf{w} \in \mathcal{W}$. We shall use this notion to describe the relation between a Krylov subspace and a desired invariant subspace $\mathcal{X}_g$ corresponding to good (or wanted) eigenvalues of $\mathbf{A}$.

The following lemma will provide a decomposition of $\mathcal{K}(\mathbf{A}, \mathbf{v}_1)$ associated with a given starting vector $\mathbf{v}_1$. This lemma provides a fundamental step towards understanding convergence in gap between $\mathcal{X}_g$ and $\mathcal{K}(\mathbf{A}, \mathbf{v}_1)$.

Lemma 5.1.

$$\mathcal{K} = \oplus_{j=1}^{N} \mathcal{K}_{k_j}(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1),$$

where $k_j \leq n_j$ is the degree of the minimal polynomial of $\mathbf{A}$ with respect to $\mathbf{P}_j \mathbf{v}_1$.

Proof. Since $\mathbf{P}_j \mathbf{v}_1$ is in the invariant subspace $\mathcal{S}_j$, we have $\mathcal{K}_\ell(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1) \subset \mathcal{S}_j$ for all ℓ. Given any $\mathbf{x} = \psi(\mathbf{A})\mathbf{v}_1 \in \mathcal{K}$, we have

$$\mathbf{x} = \psi(\mathbf{A}) \left(\sum_{j=1}^{N} \mathbf{P}_j \mathbf{v}_1 \right) = \sum_{j=1}^{N} \psi(\mathbf{A}) \mathbf{P}_j \mathbf{v}_1 \in \oplus_{j=1}^{N} \mathcal{K}_{k_j}(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1)$$

(this is a direct sum since $\mathcal{S}_i \cap \mathcal{S}_j = \{\mathbf{0}\},\ i \neq j$). To demonstrate the opposite containment, let $\mathbf{x} \in \oplus_{j=1}^{N} \mathcal{K}_{k_j}(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1)$. Then $\mathbf{x} = \sum_{j=1}^{N} \psi_j(\mathbf{A}) \mathbf{P}_j \mathbf{v}_1$ with $\deg(\psi_j) < k_j$. Let ϕ be the unique polynomial of degree $\hat{k} < k_1 + k_2 + \cdots + k_N$ that interpolates the specified Hermite data $(\lambda_j, \psi_j^{(\ell)}(\lambda_j))$ for $0 \leq \ell \leq k_j - 1$ and for $1 \leq j \leq N$. At each j, after expanding ϕ in a Taylor series about λ_j, we see that

$$\phi(\mathbf{A})\mathbf{P}_j\mathbf{v}_1 = \mathbf{X}_j\phi(\mathbf{R}_j)\mathbf{Y}_j^*\mathbf{v}_1 = \mathbf{X}_j\sum_{\ell=0}^{k_j-1}\frac{\phi_j^{(\ell)}(\lambda_j)}{\ell!}(\mathbf{R}_j-\lambda_j\mathbf{I}_{n_j})^\ell\mathbf{Y}_j^*\mathbf{v}_1,$$

since $\mathbf{X}_j(\mathbf{R}_j-\lambda_j\mathbf{I}_{n_j})^\ell\mathbf{Y}_j^*\mathbf{v}_1 = (\mathbf{A}-\lambda_j\mathbf{I})^\ell\mathbf{P}_j\mathbf{v}_1 = 0$, for $\ell \geq k_j$.

Thus, $\psi_j(\mathbf{A})\mathbf{P}_j\mathbf{v}_1 = \phi(\mathbf{A})\mathbf{P}_j\mathbf{v}_1$, since the Hermite interpolation conditions imply that the leading k_j terms of the Taylor expansion of ψ_j and ϕ about λ_j will agree. Hence

$$\mathbf{x} = \sum_{j=1}^N \psi_j(\mathbf{A})\mathbf{P}_j\mathbf{v}_1 = \sum_{j=1}^N \phi(\mathbf{A})\mathbf{P}_j\mathbf{v}_1 = \phi(\mathbf{A})\left(\sum_{j=1}^N \mathbf{P}_j\mathbf{v}_1\right) \in \mathcal{K}. \qquad \Box$$

We define $\mathbf{P}_g := \sum_{j=1}^L \mathbf{P}_j$ and $\mathbf{P}_b := \sum_{j=L+1}^N \mathbf{P}_j$, and we use the notation Ω_g and Ω_b to denote two open sets containing the good and bad eigenvalues respectively. We assume the closures of these regions are two disjoint sets with the appropriate connectedness and regularity of boundaries to make all of the contour integrals appearing below well defined. (Note: in those integrals, the factor $1/2\pi i$ has been absorbed into the $\mathrm{d}\zeta$ term.)

With polynomial restart techniques, we attempt to modify the starting vector $\mathbf{v}_1$ in a systematic way to force the invariant subspace $\mathcal{K}_k(\mathbf{A},\mathbf{v}_1)$ ever closer to a desired invariant subspace $\mathcal{X}_g$ corresponding to the desired eigenvalues λ_j, $1 \leq j \leq L$. From the spectral decomposition, we know that

$$\mathcal{X}_g = \oplus_{j=1}^L \mathcal{S}_j = \mathrm{Range}(\mathbf{P}_g),$$

where $\mathcal{S}_j = \mathrm{Range}(\mathbf{P}_j)$. We shall define the complementary space $\mathcal{X}_b := \mathrm{Range}(\mathbf{P}_b)$. From Lemma 5.1 we have

$$\mathcal{K}(\mathbf{A},\mathbf{v}_1) = \mathcal{U}_g \oplus \mathcal{U}_b,$$

where $\mathcal{U}_g := \oplus_{j=1}^L \mathcal{K}_{k_j}(\mathbf{A},\mathbf{P}_j\mathbf{v}_1)$ and $\mathcal{U}_b := \oplus_{j=L+1}^N \mathcal{K}_{k_j}(\mathbf{A},\mathbf{P}_j\mathbf{v}_1)$.

The questions we hope to answer are:

What is the gap $\delta(\mathcal{X}_g, \mathcal{K}(\mathbf{A},\mathbf{v}_1))$?
What is the gap $\delta(\mathcal{X}_g, \mathcal{K}(\mathbf{A},\hat{\mathbf{v}}_1))$ with $\hat{\mathbf{v}}_1 = \Phi(\mathbf{A})\mathbf{v}_1$?

The following discussion attempts to answer these questions.

Definition. Given a starting vector $\mathbf{v}_1$ and a selection of 'good' eigenvalues $\{\lambda_j : 1 \leq j \leq L\}$ with corresponding invariant subspace $\mathcal{X}_g$, we define the maximal reachable set $\mathcal{U}_{\max}$ to be

$$\mathcal{U}_{\max} := \mathcal{K}(\mathbf{A},\mathbf{v}_1) \cap \mathcal{X}_g.$$

It is easily seen that $\mathcal{U}_{\max}$ is invariant with respect to $\mathbf{A}$ and the following lemma will characterize this invariant subspace precisely.

Lemma 5.2. Given a starting vector $\mathbf{v}_1$ and a selection of 'good' eigenvalues $\{\lambda_j : 1 \le j \le L\}$, the maximal reachable set $\mathcal{U}_{\max}$ is

$$\mathcal{U}_{\max} = \oplus_{j=1}^{L} \mathcal{K}_{k_j}(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1),$$

and therefore

$$\mathcal{U}_{\max} = \mathcal{U}_g.$$

Proof. The proof is immediate from the characterization of $\mathcal{K}(\mathbf{A}, \mathbf{v}_1)$ and the fact that $\mathcal{K}_\ell(\mathbf{A}, \mathbf{P}_j \mathbf{v}_1) \subset \mathcal{S}_j$ for all ℓ. □

Unfortunately, there are situations where it is impossible to produce a Krylov space that contains a good approximating subspace to all of $\mathcal{X}_g$. Note that $\mathcal{U}_g \subset \mathcal{X}_g$, and that the only possibility for this containment to be proper is if $k_j < n_j$ for some $1 \le j \le L$. That is to say, at least one good eigenvalue must be derogatory. The following lemma establishes that it is impossible to converge to all of $\mathcal{X}_g$ whenever there is a derogatory eigenvalue amongst the good eigenvalues.

Lemma 5.3. Suppose $\mathcal{U}_g \subset \mathcal{X}_g$ is a proper subset of $\mathcal{X}_g$. Then

$$\delta(\mathcal{X}_g, \mathcal{K}(\mathbf{A}, \mathbf{v}_1)) \ge \frac{1}{\|\mathbf{P}_g\|}.$$

Proof. Since $\mathcal{U}_g$ is a proper subset of $\mathcal{X}_g$, there is a $\mathbf{z} \in \mathcal{X}_g$ such that $\|\mathbf{z}\| = 1$ and $\mathbf{z} \in \mathcal{U}_g^{\perp}$. Thus, for any $\mathbf{v}_g \in \mathcal{U}_g$ we must have

$$\|\mathbf{v}_g - \mathbf{z}\|^2 = \|\mathbf{v}_g\|^2 + \|\mathbf{z}\|^2 \ge \|\mathbf{z}\|^2 = 1.$$

Now, since any $\mathbf{v} \in \mathcal{K} := \mathcal{K}(\mathbf{A}, \mathbf{v}_1)$ can be written uniquely as $\mathbf{v} = \mathbf{v}_g + \mathbf{v}_b$ with $\mathbf{v}_g \in \mathcal{U}_g$ and $\mathbf{v}_b \in \mathcal{U}_b$, we have

$$\begin{aligned}
\delta(\mathcal{X}_g, \mathcal{K}(\mathbf{A}, \mathbf{v}_1)) &= \max_{\mathbf{u} \in \mathcal{X}_g} \min_{\mathbf{v} \in \mathcal{K}} \frac{\|\mathbf{v} - \mathbf{u}\|}{\|\mathbf{u}\|} \ge \min_{\mathbf{v} \in \mathcal{K}} \frac{\|\mathbf{v} - \mathbf{z}\|}{\|\mathbf{z}\|} \\
&\ge \min_{\substack{\mathbf{v}_g \in \mathcal{U}_g \\ \mathbf{v}_b \in \mathcal{U}_b}} \frac{\|\mathbf{v}_g + \mathbf{v}_b - \mathbf{z}\|}{\|\mathbf{z}\|} \ge \min_{\substack{\mathbf{v}_g \in \mathcal{U}_g \\ \mathbf{v}_b \in \mathcal{U}_b}} \frac{\|(\mathbf{v}_g - \mathbf{z}) + \mathbf{v}_b\|}{\|\mathbf{v}_g - \mathbf{z}\|} \\
&\ge \min_{\substack{\mathbf{y} \in \mathcal{X}_g \\ \mathbf{v}_b \in \mathcal{X}_b}} \frac{\|\mathbf{v}_b - \mathbf{y}\|}{\|\mathbf{y}\|} = \min_{\substack{\mathbf{y} \in \mathcal{X}_g \\ \mathbf{v}_b \in \mathcal{X}_b}} \frac{\|\mathbf{v}_b - \mathbf{y}\|}{\|\mathbf{P}_g(\mathbf{v}_b - \mathbf{y})\|} \\
&\ge \min_{\mathbf{x}} \frac{\|\mathbf{x}\|}{\|\mathbf{P}_g \mathbf{x}\|} = \frac{1}{\|\mathbf{P}_g\|}.
\end{aligned}$$

□

In this lemma, we could just as well have replaced $\mathcal{X}_g$ with any subspace $\mathcal{U}$ of $\mathcal{X}_g$ that properly contains $\mathcal{U}_g$. This justifies calling $\mathcal{U}_g$ the maximal reachable subspace. Moreover, since $\mathcal{K}(\mathbf{A}, \Phi(\mathbf{A})\mathbf{v}_1)$ is a subspace of $\mathcal{K}(\mathbf{A}, \mathbf{v}_1)$, the result also applies to all possible subspaces obtained by polynomial restarting.

The best we can hope for is to produce a Krylov space that contains an approximation to $\mathcal{U}_g$. Of course, when the dimension is sufficiently large, $\mathcal{U}_g$ will be captured exactly, since $\mathcal{U}_g \subset \mathcal{K}(\mathbf{A}, \mathbf{v}_1)$. We are interested in cases where the dimension of the Krylov space is reasonably small.

We begin with a discussion of the distance of a Krylov space of dimension ℓ from $\mathcal{U}_g$, and then introduce the consequences for restarting.

Lemma 5.4. Let $\ell \geq m = \dim\{\mathcal{U}_g\}$. Then

$$\delta(\mathcal{U}_g, \mathcal{K}_\ell(\mathbf{A}, \mathbf{v}_1)) \leq \max_{\psi} \min_{\phi} \frac{\|\phi(\mathbf{A})\mathbf{P}_b \mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g \mathbf{v}_1\|}$$

such that $\phi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1 = \psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1$ and $\deg(\phi) < \ell$, $\deg(\psi) < m$.

Proof. Since $\mathcal{U}_g = \oplus_{j=1}^{L} \mathcal{K}_{k_j}(\mathbf{A}, \mathbf{P}_j\mathbf{v}_1)$, $\mathbf{x} \in \mathcal{U}_g$ implies

$$\mathbf{x} = \sum_{j=1}^{L} \psi_{k_j}(\mathbf{A})\mathbf{P}_j\mathbf{v}_1 = \psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1,$$

where ψ is the unique polynomial of degree less than m that interpolates the Hermite data defining ψ_{k_j}, $1 \leq j \leq L$. Also, $\mathbf{v} \in \mathcal{K}_\ell(\mathbf{A}, \mathbf{v}_1)$ implies $\mathbf{v} = \phi(\mathbf{A})\mathbf{v}_1$ with $\deg(\phi) < \ell$. Thus

$$\begin{aligned}
\delta(\mathcal{U}_g, \mathcal{K}_\ell(\mathbf{A}, \mathbf{v}_1)) &= \max_{\psi} \min_{\phi} \frac{\|\phi(\mathbf{A})\mathbf{v}_1 - \psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \\
&= \max_{\psi} \min_{\phi} \frac{\|[\phi(\mathbf{A}) - \psi(\mathbf{A})]\mathbf{P}_g\mathbf{v}_1 + \phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \\
&\leq \max_{\psi} \min_{\phi} \frac{\|\phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|},
\end{aligned}$$

where the final inequality is obtained by restricting ϕ to satisfy the Hermite interpolation data defining ψ on λ_j for $1 \leq j \leq L$. □

We wish to refine this estimate into a more quantitative one. Recall $m = \dim\{\mathcal{U}_g\} = \sum_{j=1}^{L} k_j$ and $\ell \geq m$. Define $\alpha(\tau)$ to be the minimal polynomial of $\mathbf{A}$ with respect to $\mathbf{P}_g\mathbf{v}_1$. It is straightforward to show $\alpha(\tau) = \prod_{j=1}^{L}(\tau - \lambda_j)^{k_j}$. Moreover, any polynomial ϕ of degree $\ell - 1 \geq m - 1$ satisfying the constraint of Lemma 5.4 must be of the form

$$\phi(\tau) = \psi(\tau) + \hat{\phi}(\tau)\alpha(\tau).$$

Intuitively, this means that the matrix $\phi(\mathbf{A}) - \psi(\mathbf{A})$ must annihilate $\mathcal{U}_g$. We have the following result.

Corollary 5.5.

$$\delta(\mathcal{U}_g, \mathcal{K}_\ell(\mathbf{A}, \mathbf{v}_1)) \leq \max_{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|=1} \min_{\hat{\phi}} \|[\psi(\mathbf{A}) + \hat{\phi}(\mathbf{A})\alpha(\mathbf{A})]\mathbf{P}_b\mathbf{v}_1\|.$$

Thus, our gap estimate amounts to a question of how well a polynomial $\hat{\phi}$ of degree $\ell - m$ can approximate the rational function $\frac{\psi(\tau)}{\alpha(\tau)}$ over certain regions of the complex plane and, in particular, over the region Ω_b.

We can easily verify

$$\|[\psi(\mathbf{A}) + \hat{\phi}(\mathbf{A})\alpha(\mathbf{A})]\mathbf{P}_b\mathbf{v}_1\| = \left\| \oint_{\partial\Omega_b} [\psi(\zeta) + \hat{\phi}(\zeta)\alpha(\zeta)](\zeta\mathbf{I} - \mathbf{A})^{-1}\mathbf{P}_b\mathbf{v}_1 \, \mathrm{d}\zeta \right\|,$$

where (as specified above) Ω_b includes the bad eigenvalues and excludes the good ones (with sufficient regularity conditions on connectedness and smoothness of the boundary).

The previous discussion gives a qualitative idea of how the bounds will be obtained, but does not really lead to a concrete bound since ψ may be arbitrarily large.

5.2. Bounding $\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|$ from below

We first consider the case that there is just one wanted eigenvalue λ_1 and that $\alpha(\tau) = (\tau - \lambda_1)^{k_1}$ is the minimal polynomial of $\mathbf{A}$ with respect to $\mathbf{P}_g\mathbf{v}_1$. We may conclude that $\|(\mathbf{A} - \lambda_1\mathbf{I})^j\mathbf{P}_g\mathbf{v}_1\| > 0$ for $0 \leq j < k_1$. Given ψ of degree less than k_1, set

$$\psi(\tau) = \hat{\psi}(\tau)(\tau - \lambda_1)^j, \ (j < k_1),$$

where we assume that $\hat{\psi}(\lambda_1) = 1$ since the numerator and denominator in Lemma 5.4 may be simultaneously scaled by the same nonzero constant. Now, let $\mathbf{\Lambda}_\epsilon = \{\zeta \in \mathbb{C} : \|(\zeta\mathbf{I} - \mathbf{A})^{-1}\| \geq \frac{1}{\epsilon}\}$. The set $\mathbf{\Lambda}_\epsilon$ is called the ϵ-pseudo-spectrum of $\mathbf{A}$ (Trefethen 1992, 1999). This is one of several equivalent descriptions. The boundaries of this family of sets are level curves of the function $\|(\zeta\mathbf{I} - \mathbf{A})^{-1}\|$ and these are called lemniscates. Let ϵ be sufficiently small that there is a connected component of $\mathbf{\Lambda}_\epsilon$ denoted by Ω_g that contains λ_1 and no other eigenvalue of $\mathbf{A}$. Then $\|(\zeta\mathbf{I} - \mathbf{A})^{-1}\| = 1/\epsilon$ on $\partial\Omega_g$ and since $\hat{\psi}(\lambda_1) = 1$, we may take ϵ sufficiently small to ensure $|\hat{\psi}(\zeta)| > 1/2$ on $\partial\Omega_g$. Then

$$\begin{aligned}
\|(\mathbf{A} - \lambda_1\mathbf{I})^j\mathbf{P}_g\mathbf{v}_1\| &= \left\| \left(\oint_{\partial\Omega_g} \frac{1}{\hat{\psi}(\zeta)} (\zeta\mathbf{I} - \mathbf{A})^{-1} \, \mathrm{d}\zeta \right) \hat{\psi}(\mathbf{A})\mathbf{P}_g(\mathbf{A} - \lambda_1\mathbf{I})^j\mathbf{P}_g\mathbf{v}_1 \right\| \\
&\leq \left\| \oint_{\partial\Omega_g} \frac{1}{\hat{\psi}(\zeta)} (\zeta\mathbf{I} - \mathbf{A})^{-1} \, \mathrm{d}\zeta \right\| \|\hat{\psi}(\mathbf{A})(\mathbf{A} - \lambda_1\mathbf{I})^j\mathbf{P}_g\mathbf{v}_1\|.
\end{aligned}$$

Thus,

$$\begin{aligned}
\frac{\|(\mathbf{A} - \lambda_1\mathbf{I})^j\mathbf{P}_g\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} &\leq \left(\max_{\xi \in \partial\Omega_g} \frac{1}{|\hat{\psi}(\xi)|} \right) \oint_{\partial\Omega_g} \|(\zeta\mathbf{I} - \mathbf{A})^{-1}\| |\, \mathrm{d}\zeta| \\
&\leq \frac{\mathcal{L}_g}{\pi\epsilon} =: C_1,
\end{aligned}$$

where $\mathcal{L}_g$ is the length of the boundary of Ω_g.

With a little more work, this argument may be extended to L eigenvalues with $\psi(\tau) = \hat{\psi}(\tau)\hat{\alpha}(\tau)$ where $\hat{\alpha}(\tau) = \prod_{j=1}^{L}(\tau - \lambda_j)^{\ell_j}$ with $\ell_j < k_j$. In this case Ω_g is the union of the disjoint ϵ-lemniscates enclosing the good eigenvalues, and $\psi(\zeta) = \hat{\psi}(\zeta)\hat{\alpha}(\zeta)$, where $\hat{\psi}$ has been normalized to have absolute value greater than or equal to one at all of the good eigenvalues. As before, we assume that ϵ is sufficiently small to ensure that $|\hat{\psi}(\zeta)| > 1/2$ on $\partial\Omega_g$. Then, the bound becomes

$$\frac{\|\hat{\alpha}(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \le \frac{\mathcal{L}_g}{\pi\epsilon} =: C_1.$$

5.3. Bounding $\|\phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|$ from above

We impose the restriction $\phi(\tau) = \psi(\tau) + \hat{\phi}(\tau)\alpha(\tau)$ and consider

$$\begin{aligned}\phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1 &= [\psi(\mathbf{A}) + \hat{\phi}(\mathbf{A})\alpha(\mathbf{A})]\mathbf{P}_b\mathbf{v}_1 \\ &= \left(\oint_{\partial\Omega_b}\left[\frac{\psi(\zeta)}{\alpha(\zeta)} + \hat{\phi}(\zeta)\right](\zeta\mathbf{I} - \mathbf{A})^{-1}\,\mathrm{d}\zeta\right)\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1,\end{aligned}$$

which is valid since $\frac{\psi(\zeta)}{\alpha(\zeta)}$ is analytic on Ω_b. Hence,

$$\begin{aligned}\|\phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\| &\le \left\|\oint_{\partial\Omega_b}\left[\frac{\psi(\zeta)}{\alpha(\zeta)} + \hat{\phi}(\zeta)\right](\zeta\mathbf{I} - \mathbf{A})^{-1}\,\mathrm{d}\zeta\right\|\|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\| \\ &\le \max_{\zeta\in\partial\Omega_b}\left|\frac{\psi(\zeta)}{\alpha(\zeta)} + \hat{\phi}(\zeta)\right|\oint_{\partial\Omega_b}\|(\zeta\mathbf{I} - \mathbf{A})^{-1}\|\,|\mathrm{d}\zeta|\,\|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|.\end{aligned}$$

5.4. Gap estimates for polynomial restarting

We now consider the possibilities for achieving convergence in gap through polynomial restarting. This will be analysed by revising the previous estimates when we replace $\mathbf{v}_1$ with $\hat{\mathbf{v}}_1 = \Phi(\mathbf{A})\mathbf{v}_1$, where Φ is the aggregate restart polynomial. We shall assume that all of the roots of Φ are in $\mathbb{C}\ \Omega_b$. In this case we have

$$\begin{aligned}\delta(\mathcal{U}_g, \mathcal{K}_\ell(\mathbf{A}, \hat{\mathbf{v}}_1)) &= \max_{\psi}\min_{\phi}\frac{\|\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{v}_1 - \psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \\ &= \max_{\psi}\min_{\phi}\frac{\|[\phi(\mathbf{A})\Phi(\mathbf{A}) - \psi(\mathbf{A})]\mathbf{P}_g\mathbf{v}_1 + \phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \\ &\le \max_{\psi}\min_{\phi}\frac{\|\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|},\end{aligned}$$

if $\phi\cdot\Phi$ is restricted to satisfy the Hermite interpolation data defining ψ on λ_j for $1 \le j \le L$.

Motivated by the arguments above, we put

$$\phi(\tau)\Phi(\tau) = \psi(\tau) + \Psi(\tau)\alpha(\tau).$$

This is accomplished by requiring ϕ to be specified so that $\phi \cdot \Phi$ does indeed satisfy the Hermite interpolation data defining ψ on λ_j for $1 \leq j \leq L$. (This is possible since Φ has no zeros in Ω_g.)

Once we have ϕ defined, observe that $\Phi(\tau_j) = 0$ will imply that

$$\Psi(\tau_j) = -\frac{\psi(\tau_j)}{\alpha(\tau_j)}.$$

Hence, Ψ interpolates $-\frac{\psi}{\alpha}$ at each root τ_j of Φ. (This is also true at the roots of ϕ, but we have no control over the placement of those.) Note: this interpolation property is automatic and nothing need be done to enforce it.

Again, converting to integral form gives

$$\begin{aligned}\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1 &= [\psi(\mathbf{A}) + \Psi(\mathbf{A})\alpha(\mathbf{A})]\mathbf{P}_b\mathbf{v}_1 \\ &= \left(\oint_{\partial\Omega_b} \left[\frac{\psi(\zeta)}{\alpha(\zeta)} + \Psi(\zeta)\right](\zeta\mathbf{I} - \mathbf{A})^{-1}\,\mathrm{d}\zeta\right)\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1,\end{aligned}$$

and this allows us to obtain the estimate

$$\begin{aligned}&\|\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\| \\ &\quad \leq \left\| \oint_{\partial\Omega_b} \left[\frac{\psi(\zeta)}{\alpha(\zeta)} + \Psi(\zeta)\right](\zeta\mathbf{I} - \mathbf{A})^{-1}\,\mathrm{d}\zeta\right\| \|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\| \\ &\quad \leq \max_{\zeta\in\partial\Omega_b} \left|\frac{\psi(\zeta)}{\alpha(\zeta)} + \Psi(\zeta)\right| \oint_{\partial\Omega_b} \|(\zeta\mathbf{I} - \mathbf{A})^{-1}\|\,|\mathrm{d}\zeta| \|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|.\end{aligned}$$

If we assume $\Omega_g \cup \Omega_b$ consists of the ϵ-pseudospectrum of $\mathbf{A}$ with ϵ sufficiently small that the closures of these sets do not intersect, then $\|(\zeta\mathbf{I} - \mathbf{A})^{-1}\| = \frac{1}{\epsilon}$ for $\zeta \in \partial\Omega_b$, and we obtain

$$\|\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\| \leq \max_{\zeta\in\partial\Omega_b} \left|\frac{\psi(\zeta)}{\alpha(\zeta)} + \Psi(\zeta)\right| \frac{\mathcal{L}_b}{2\pi\epsilon} \|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|,$$

where $\mathcal{L}_b$ is the length of the boundary of Ω_b.

Since we are free to choose the roots of Φ, we should be able to make this estimate arbitrarily and uniformly small. The key to this will be the selection of points that have desirable asymptotic approximation properties with respect to interpolation of a given rational function on Ω_b at an increasing number of points. Leja points (and also Fejér or Fekete points) are known to have such properties but they are expensive to compute. A more attractive option is the use of so-called fast Leja points, introduced in Baglama, Calvetti and Reichel (1998). Fast Leja points give almost the same interpolation behaviour as Leja points but they are efficiently computed (as shown in Baglama *et al.* (1998)). The construction amounts to a recursively defined distribution of the points on $\partial\Omega_b$ in a way that is nearly optimal with respect to asymptotic interpolation properties. There are several additional properties that make these points very attractive computationally (see Baglama *et al.* (1998)).

There is no asymptotic rate of convergence available for fast Leja points, but there is one for Leja points that does ensure a linear rate of convergence for our application. Let us suppose for the moment that ψ is a fixed polynomial of degree $\ell - 1$ and α is as specified above. The following result may be found in Gaier (1987) and in related papers (Reichel 1990, Fischer and Reichel 1989).

Theorem 5.6. Assume that $\partial\Omega_b$ is a Jordan curve. Let $\mathcal{G}(\omega)$ be the conformal mapping from the exterior of the unit disk to the exterior of Ω_b, such that $\mathcal{G}(\infty) = \infty$ and $\mathcal{G}'(\infty) > 0$ (guaranteed to exist by the Riemann mapping theorem). We want to approximate $f(\zeta) = \frac{\psi(\zeta)}{\alpha(\zeta)}$ on Ω_b. Since α has all zeros outside Ω_b, there is a circle with radius $\rho > 1$ and centre at the origin, such that its image $\mathcal{C}$ under $\mathcal{G}$ goes through a zero of α and there is no zero of α in the interior of the curve $\mathcal{C}$. Let q_M be the polynomial of degree $< M$ that interpolates f at M Leja (Fejér or Fekete) points on the boundary of Ω_b. Then

$$\limsup_{M\to\infty} \max_{\zeta\in\Omega_b} |f(\zeta) - q_M(\zeta)|^{\frac{1}{M}} = \frac{1}{\rho}.$$

Thus we expect a linear rate of convergence with ratio ρ^{-1}. If we apply p shifts at a time the convergence factor should be ρ^{-p}. The convergence is, of course, faster for larger ρ: *i.e.*, as the distance of the zeros of α from Ω_b increases, so does ρ, and the convergence is correspondingly faster. Recalling that the zeros of α are the desired eigenvalues, this confirms and makes precise the intuitive notion that convergence should be faster when the wanted eigenvalues are well separated from the rest of the spectrum.

The final convergence result will be of the form

$$\begin{aligned}\delta(\mathcal{U}_g, \mathcal{K}_\ell(\mathbf{A}, \hat{\mathbf{v}}_1)) &\le \max_{\psi} \min_{\phi} \frac{\|\phi(\mathbf{A})\Phi(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\psi(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|} \\ &\le \left(\frac{1}{\rho}\right)^M C_0C_1C_2 \max_{\hat{\alpha}} \frac{\|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\hat{\alpha}(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|},\end{aligned}$$

where $C_1 = \frac{\mathcal{L}_g}{\pi\epsilon}$, $C_2 = \frac{\mathcal{L}_b}{2\pi\epsilon}$, and where $\hat{\alpha}(\tau) = \prod_{j=1}^{L}(\tau - \lambda_j)^{\ell_j}$ with $\ell_j < k_j$. The positive constant C_0 is associated with converting the *lim-sup* statement to a convergence rate. The integer $M = \deg(\Phi) = \nu p$ if there have been ν restarts of degree p.

These terms have very natural interpretations. In particular, ρ is determined by the distance of the good eigenvalues $\lambda_j,\ \ 1 \le j \le L$ from the set Ω_b enclosing the bad eigenvalues. The constants C_1 and C_2 are related to the nearness to nonnormality through the behaviour of the ϵ-pseudospectra. Finally, the ratio $\frac{\|\alpha(\mathbf{A})\mathbf{P}_b\mathbf{v}_1\|}{\|\hat{\alpha}(\mathbf{A})\mathbf{P}_g\mathbf{v}_1\|}$ reflects the influence of bias in the starting vector towards $\mathcal{U}_g$. A pleasing consequence of this term is that, whenever $\mathbf{v}_1 \in \mathcal{U}_g$, then there is termination as soon as $\ell \ge m$ in exact arithmetic.

Certain lemniscates of the ϵ-pseudospectra will form the boundary of Ω_b, and hence (unless they just touch) this boundary will be a union of Jordan curves. In certain cases, we can obtain concrete estimates by replacing Ω_b with another set that encloses all of the bad eigenvalues, and with a positive distance from Ω_g. If this new set can be constructed so that the integrals can be calculated or estimated, then actual convergence rates follow. In practice, it is unusual to have advanced knowledge of such a set. In the symmetric case, such sets are ϵ-balls centred at the eigenvalues, and this leads to containment intervals on the real line. There is a method for constructing Leja points for an adaptively defined containment interval. This has been quite successful, as demonstrated in Baglama *et al.* (1996). Exact shifts tend to discover such regions adaptively. As we have seen in prior examples, they distribute themselves near the boundary of the adaptively discovered containment region. This is one heuristic reason why exact shifts seem to be successful in many cases. See Beattie *et al.* (2001) for a convincing computational example of this.

6. Subspace iteration methods

There is another generalization of the power method that is perhaps more straightforward than Krylov subspace projection. This is the generalized power method or subspace iteration. It treats a block of vectors simultaneously in a direct analogy to the power method. In Algorithm 6 a shift-invert variant of this method is described.

```
Factor $\mathbf{VR} = \mathbf{W}$ (with $\mathbf{W} \in \mathbb{C}^{n\times k}$ arbitrary);
Set $\mathbf{H} = 0$;
while ($\|\mathbf{AV} - \mathbf{VH}\| > \mathrm{tol}\|\mathbf{H}\|$),
    $\mu$ = Select_shift($\mathbf{H}$);
    Solve $(\mathbf{A} - \mu\mathbf{I})\mathbf{W} = \mathbf{V}$;
    Factor $[\mathbf{V}_+, \mathbf{R}] = \mathrm{qr}(\mathbf{W})$;
    $\mathbf{H} = \mathbf{V}_+^*\mathbf{V}\mathbf{R}^{-1} + \mu\mathbf{I}$;
    $\mathbf{V} \leftarrow \mathbf{V}_+$;
end
```

Algorithm 6. Generalized shifted inverse power method

Noting that $\mathbf{H} = \mathbf{V}^*\mathbf{AV}$ in Algorithm 6, it is evident that the Ritz pairs $(\mathbf{x}, \theta)$ may be obtained from the eigensystem of $\mathbf{H}$ just as in the Krylov setting. However, in this case the subspace $\mathcal{S} = \mathrm{Range}(\mathbf{V})$ will be dominated by eigenvector directions corresponding to the eigenvalues nearest to the selected shifts μ_j. The stopping rule can be modified so that additional matrix-vector products to obtain $\mathbf{AV}$ are not explicitly required. Also, in practice, it will most likely be more reliable to compute the final value of $\mathbf{H}$ (after convergence) by computing $\mathbf{V}^*(\mathbf{AV})$ directly.

Typically a single shift μ is selected and a single sparse direct factorization of $\mathbf{A} - \mu\mathbf{I}$ is computed initially and re-used to solve the systems

$$(\mathbf{A} - \mu\mathbf{I})\mathbf{W} = \mathbf{V}$$

repeatedly as needed. In this case, it is easily seen that the result on convergence is a partial Schur decomposition,

$$\mathbf{AV} = \mathbf{V}(\mathbf{R}^{-1} + \mu\mathbf{I}).$$

When $k = n$ this iteration becomes the well-known and very important shifted QR iteration. To see this, suppose an initial orthogonal similarity transformation of $\mathbf{A}$ to upper Hessenberg form has been made so that

$$\mathbf{AV} = \mathbf{VH} \quad \text{with} \quad \mathbf{V}^*\mathbf{V} = \mathbf{I}, \quad \mathbf{H} \quad \text{upper Hessenberg.}$$

If $\mathbf{H} = \mathbf{QR}$ is the QR factorization of $\mathbf{H}$, then $\mathbf{W} = (\mathbf{VQ})\mathbf{R}$ is the QR factorization of $\mathbf{W} = \mathbf{AV}$. Moreover,

$$\begin{aligned}(\mathbf{A} - \mu\mathbf{I})(\mathbf{VQ}) &= (\mathbf{VQ})(\mathbf{RQ}),\\ \mathbf{AV}_+ &= \mathbf{V}_+\mathbf{H}_+ \quad \text{with} \quad \mathbf{V}_+^*\mathbf{V}_+ = \mathbf{I}, \quad \mathbf{H}_+ = \mathbf{RQ} + \mu\mathbf{I}.\end{aligned}$$

Of course, the amazing thing is that $\mathbf{H}_+$ remains upper Hessenberg if $\mathbf{H}$ is originally upper Hessenberg. Moreover, the QR factorization of $\mathbf{H}$ by Givens' method and the associated updating $\mathbf{V}_+ = \mathbf{VQ}$ requires $O(n^2)$ flops rather than $O(n^3)$ for a dense QR factorization.

The important observation to make with this iteration is that the construction of the subspace is divorced from the construction of Ritz vectors. Therefore, the system $(\mathbf{A} - \mu\mathbf{I})\mathbf{W} = \mathbf{V}$ could just as well be solved (approximately) with an iterative method. The projected matrix $\mathbf{H}$ would then be obtained directly by forming $\mathbf{H} \leftarrow \mathbf{V}_+^*\mathbf{AV}_+$. The Krylov structure would be lost with this approach, but there are trade-offs.

A downside to abandoning the Krylov structure is a loss of efficiency in obtaining Ritz approximations and associated error estimates directly from $\mathbf{H}$. Also, certain very powerful polynomial approximation properties are lost. However, there are some significant advantages.

- There is the possibility of admitting approximate solutions to the block linear system $(\mathbf{A} - \mu\mathbf{I})\mathbf{W} = \mathbf{V}$. Other than the effect on convergence, there is no set accuracy requirement for these solves. This is in contrast to the Krylov setting, where important theoretical properties are lost if these solves are not accurate enough.

- If a sequence of closely related problems is being solved, as in a parameter study, the entire subspace basis from the previous problem can be used as the initial basis for the next problem. In the (single-vector) Krylov setting we must be content with a linear combination of the previous basis vectors (or Ritz vectors) to form a single starting vector for the next problem.
- If an iterative method is used to solve $(\mathbf{A} - \mu\mathbf{I})\mathbf{w} = \mathbf{v}$ approximately, then several matrix-vector products are performed for each access to the matrix $\mathbf{A}$. This can be quite important on high-performance architectures where it is desirable to perform as many floating point operations as possible per each memory access.
- It is possible to be very general in constructing vectors to adjoin to the subspace. Schemes may be devised that do not attempt to solve the shift-invert equations directly, but instead attempt to construct defect corrections as vectors to adjoin to the subspace. Davidson's method (Davidson 1975) and its variants are based on this idea.

6.1. Davidson's method

Davidson's method has been a mainstay in computational chemistry, where it is generally preferred over the Lanczos method. Typically, *ab initio* calculations in chemistry result in large symmetric positive definite matrices, which are strongly diagonally dominant.

Davidson's method attempts to exploit that structure. Given a subspace $\mathcal{S}_k = \text{Range}(\mathbf{V}_k)$ of dimension k with orthogonal basis matrix $\mathbf{V}_k$ and a selected Ritz value $\theta \in \sigma(\mathbf{V}_k^*\mathbf{A}\mathbf{V}_k)$ with corresponding Ritz vector $\hat{\mathbf{x}}$, the strategy is to expand the space with a residual defect correction designed to improve the selected Ritz value. In the following discussion one should regard $\hat{\mathbf{x}}$ as the current approximation to an eigenvector $\mathbf{x}$ and θ as the current approximation to the corresponding eigenvalue θ.

If λ is the closest eigenvalue to θ and $\mathbf{x}$ is a corresponding eigenvector, putting $\mathbf{x} = \hat{\mathbf{x}} + \mathbf{z}$ and $\lambda = \theta + \delta$ and expanding gives the standard second-order perturbation equation

$$\begin{aligned} (\mathbf{A} - \theta\mathbf{I})\mathbf{z} &= -(\mathbf{A} - \theta\mathbf{I})\hat{\mathbf{x}} + \hat{\mathbf{x}}\delta + \mathbf{z}\delta && (6.1)\\ &= -\mathbf{r} + \hat{\mathbf{x}}\delta + \mathcal{O}(\|\mathbf{z}\delta\|) && (6.2)\\ &\approx -\mathbf{r} + \hat{\mathbf{x}}\delta. && (6.3) \end{aligned}$$

Typically, this second-order residual correction equation is completed by forcing a condition such as $\mathbf{z}^*\hat{\mathbf{x}} = 0$. Davidson (1975) chooses to approximate $\mathbf{D}_A - \theta\mathbf{I} \approx \mathbf{A} - \theta\mathbf{I}$ on the left side and ignore both the first- and second-order terms on the right side, using the equation

$$(\mathbf{D}_A - \theta\mathbf{I})\mathbf{z} = -\mathbf{r} \quad \text{where} \quad \mathbf{r} = (\mathbf{A} - \theta\mathbf{I})\hat{\mathbf{x}}$$

to obtain an approximate residual correction step $\mathbf{z}$. A novelty of the Davidson approach was to orthogonalize $\mathbf{z}$ against the existing basis set to obtain a new basis vector $\mathbf{v}_{k+1}$ in the direction $(\mathbf{I}-\mathbf{V}_k\mathbf{V}_k^*)\mathbf{z}$, and expand the space to $\mathcal{S}_{k+1} = \text{Range}(\mathbf{V}_{k+1})$ where $\mathbf{V}_{k+1} = [\mathbf{V}_k, \mathbf{v}_{k+1}]$. A new Ritz value and vector are obtained from the updated space, and then this process is repeated until a storage limit is reached and the method is restarted.

This method has been quite successful in finding dominant eigenvalues of strongly diagonally dominant matrices. Evidently, from the second-order expansion and as suggested in Davidson (1993), this scheme is related to a Newton–Raphson iteration, and this has been used as a heuristic to explain its fast convergence.

Numerical analysts have attempted to explain the success of this approach by viewing $(\mathbf{D}_A - \theta\mathbf{I})^{-1}$ as a preconditioner, or as an approximation to $(\mathbf{A} - \theta\mathbf{I})^{-1}$. With this interpretation, improvements to Davidson's method have been attempted through the introduction of more sophisticated preconditioners (Crouzeix, Philippe and Sadkane 1994, Morgan 1991, Morgan and Scott 1993). However, a perplexing aspect of this interpretation has been that the ultimate preconditioner, namely $(\mathbf{A}-\theta\mathbf{I})^{-1}$, would just result in $\mathbf{z} = \hat{\mathbf{x}}$ and would not expand the search space.

6.2. The Jacobi–Davidson method

Progress towards improving on Davidson was finally made after recognizing that the correction should be restricted to the orthogonal complement of the existing space. This notion follows almost directly from reconsidering the second-order correction equation (6.1) and completing the equations by forcing the correction $\mathbf{z}$ to be orthogonal to $\hat{\mathbf{x}}$. This may be accomplished by forming a bordered set of equations or by explicit projection. Multiplying on the left of (6.1) by the projection $(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)$ and requiring $\hat{\mathbf{x}}^*\mathbf{z} = 0$ gives

$$\begin{aligned}(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)(\mathbf{A} - \theta\mathbf{I})(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{z} &= (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)(-\mathbf{r} + \hat{\mathbf{x}}\delta + \mathcal{O}(\|\mathbf{z}\delta\|)) && (6.4)\\ &= -\mathbf{r} + \mathcal{O}(\|\mathbf{z}\delta\|) && (6.5)\\ &\approx -\mathbf{r}. && (6.6)\end{aligned}$$

The second equality follows from the observation that $\theta = \hat{\mathbf{x}}^*\mathbf{A}\hat{\mathbf{x}}$ is a Rayleigh quotient, and thus $\mathbf{r} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{x}\theta = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{A}\mathbf{x}$.

This formulation actually results in a second-order correction $\mathbf{z} = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{z}$ that is orthogonal to $\hat{\mathbf{x}}$. The coefficient matrix $(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)(\mathbf{A} - \theta\mathbf{I})(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)$ is indeed singular, but the linear system is consistent and offers no fundamental difficulty to an iterative method. Moreover, if θ approximates a simple eigenvalue that is moderately separated from the rest of the spectrum of $\mathbf{A}$, this system is likely to be better conditioned than one involving $\mathbf{A} - \theta\mathbf{I}$ as a coefficient matrix, since the nearly singular subspace has been projected out.

Now, the Davidson idea can be fully realized. The projected correction equation (6.4) is solved iteratively. Typically, this is done with a preconditioned iterative method for linear systems. Then the correction is used as with the original Davidson idea to expand the search space. Note that one step of the preconditioned GMRES method using $\mathbf{D}_A - \theta\mathbf{I}$ as a preconditioner would result in Davidson's method.

This approach due to Sleijpen and van der Vorst (1995) is called the Jacobi–Davidson (JD) method. The method applied to a symmetric $\mathbf{A} = \mathbf{A}^*$ is outlined in Algorithm 7. The hat notation $\hat{\mathbf{x}}$ is dropped in that description and $\mathbf{x}$ is the current approximate eigenvector. The update to obtain $\mathbf{H}_k$ from $\mathbf{H}_{k-1}$ is just slightly more complicated for nonsymmetric $\mathbf{A}$.

```
Set $\mathbf{x} = \mathbf{v}_1 = \mathbf{v}/||\mathbf{v}||_2$ for some initial guess $\mathbf{v}$;
$\mathbf{w} = \mathbf{A}\mathbf{v}_1$, $\theta = \mathbf{H}(1,1) = [\mathbf{v}_1^*\mathbf{w}]$, $\mathbf{r} = \mathbf{w} - \theta\mathbf{x}$;
    while $||\mathbf{r}||_2 > \varepsilon$
        Solve (approximately) for $\mathbf{z} \perp \mathbf{x}$:
            $(\mathbf{I} - \mathbf{x}\mathbf{x}^*)(\mathbf{A} - \theta\mathbf{I})(\mathbf{I} - \mathbf{x}\mathbf{x}^*)\mathbf{z} = -\mathbf{r}$;
        $\mathbf{c} = \mathbf{V}_{k-1}^*\mathbf{z}$ ; $\mathbf{z} = \mathbf{z} - \mathbf{V}_{k-1}\mathbf{c}$;
        $\mathbf{v}_k = \mathbf{z}/||\mathbf{z}||_2$; $\mathbf{V}_k = [\mathbf{V}_{k-1}, \mathbf{v}_k]$;
        $\mathbf{w} = \mathbf{A}\mathbf{v}_k$;

        $\begin{bmatrix} \mathbf{h} \\ \alpha \end{bmatrix} = \mathbf{V}_k^*\mathbf{w}$ ; $\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{h} \\ \mathbf{h}^* & \alpha \end{bmatrix}$;

        Compute $\mathbf{H}_k\mathbf{y} = \mathbf{y}\theta$;
        ($\theta$ the largest eigenvalue of $\mathbf{H}_k$, $||\mathbf{y}||_2 = 1$)
        $\mathbf{x} \leftarrow \mathbf{V}_k\mathbf{y}$;
        $\mathbf{r} = \mathbf{A}\mathbf{x} - \mathbf{x}\theta$;
    end
```

Algorithm 7. The Jacobi–Davidson method for $\lambda_{\max}(\mathbf{A})$ with $\mathbf{A} = \mathbf{A}^*$

There are several ways to approximately solve the correction equation. Returning to the second-order expansion (6.1),

$$(\mathbf{A} - \theta\mathbf{I})\mathbf{z} = -\mathbf{r} + \hat{\mathbf{x}}\delta,$$

to get $\mathbf{z}$ orthogonal to $\hat{\mathbf{x}}$, choose

$$\delta = \frac{\hat{\mathbf{x}}^*(\mathbf{A} - \theta\mathbf{I})^{-1}\mathbf{r}}{\hat{\mathbf{x}}^*(\mathbf{A} - \theta\mathbf{I})^{-1}\hat{\mathbf{x}}}.$$

If $(\mathbf{A} - \theta\mathbf{I})$ is replaced with a preconditioner $\mathbf{K}$, then we set

$$\mathbf{z} = -\mathbf{K}^{-1}\mathbf{r} + \mathbf{K}^{-1}\hat{\mathbf{x}} \quad \text{with} \quad \delta = \frac{\hat{\mathbf{x}}^*\mathbf{K}^{-1}\mathbf{r}}{\hat{\mathbf{x}}^*\mathbf{K}^{-1}\hat{\mathbf{x}}}.$$

If the basis vectors $\mathbf{V}_k$ are not retained and $\mathbf{z}$ is not orthogonalized against them, this becomes the method proposed by Olsen, Jørgensen and Simons (1990).

If this correction equation is to be solved approximately with a preconditioned iterative method, care must be taken to obtain efficiency. Left-preconditioning can be applied efficiently, and it is common to have a left preconditioner $\mathbf{K}_o$ for $\mathbf{A}$ available (*e.g.*, to solve linear systems required to track the dynamics). We can then take $\mathbf{K} := \mathbf{K}_o - \mu\mathbf{I}$ as a preconditioner for $\mathbf{A} - \mu\mathbf{I}$, where μ is a value reasonably close to the desired eigenvalue. Of course, it is possible to update $\mu = \theta_k$ at each JD iteration, but the cost of construction and factorization of a new preconditioner for each value of θ_k may overcome the gains from accelerated convergence. To be effective, the preconditioner should be restricted to a subspace that is orthogonal to $\hat{\mathbf{x}}$. Thus, it is desirable to work with the restricted preconditioner

$$\widetilde{\mathbf{K}} := (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{K}(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*).$$

This is likely to be a good preonditioner for the restricted operator $\widetilde{\mathbf{A}} := (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)(\mathbf{A} - \theta\mathbf{I})(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)$. If we use a Krylov subspace iteration method for solving

$$\widetilde{\mathbf{A}}\mathbf{z} = -\mathbf{r}$$

that is initialized with $\mathbf{z}_0 = 0$, then all vectors occurring in the iterative solution process will be automatically orthogonal to $\hat{\mathbf{x}}$.

Typically, the iterative solver will require repeated evaluation of expressions like

$$\mathbf{w} = \widetilde{\mathbf{K}}^{-1}\widetilde{\mathbf{A}}\mathbf{v}$$

for vectors $\mathbf{v}$ generated during the iteration.

Since $\hat{\mathbf{x}}^*\mathbf{v} = 0$, we first compute $\mathbf{y} = (\mathbf{A} - \theta\mathbf{I})\mathbf{v}$. Then we have to solve $\mathbf{w} \perp \hat{\mathbf{x}}$ from $\widetilde{\mathbf{K}}\mathbf{w} = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{y}$. This amounts to solving

$$(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{K}(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{w} = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{y}.$$

Observe that this equation will be satisfied by $\mathbf{w}$ if we are able to solve

$$(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{K}\mathbf{w} = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{y} \quad \text{with} \quad \mathbf{w}^*\hat{\mathbf{x}} = 0.$$

This is easily accomplished by solving the bordered equation

$$\begin{bmatrix} \mathbf{K} & \hat{\mathbf{x}} \\ \hat{\mathbf{x}}^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \delta \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ 0 \end{bmatrix}.$$

From this equation, it follows that

$$\text{(i)}\quad \mathbf{w}^*\hat{\mathbf{x}} = 0 \qquad \text{and} \qquad \text{(ii)}\quad \mathbf{K}\mathbf{w} = \mathbf{y} - \hat{\mathbf{x}}\delta.$$

Hence,

$$(\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{K}\mathbf{w} = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)(\mathbf{y} - \hat{\mathbf{x}}\delta) = (\mathbf{I} - \hat{\mathbf{x}}\hat{\mathbf{x}}^*)\mathbf{y},$$

with $\mathbf{w}^*\hat{\mathbf{x}} = 0$ as required.

This block system need not be formed explicitly. Block elimination will give the solution through the following steps:

```
Given $\mathbf{y} = (\mathbf{A} - \mu\mathbf{I})\mathbf{v}$;
    Solve $\mathbf{K}[\mathbf{t}_y\ \mathbf{t}_x] = [\mathbf{y}\ \hat{\mathbf{x}}]$;
    Set $\delta = \frac{\hat{\mathbf{x}}^*\mathbf{t}_y}{\hat{\mathbf{x}}^*\mathbf{t}_x}$;
    Set $\mathbf{w} = \mathbf{t}_y - \mathbf{t}_x\delta$.
```

Since we are interested in solving this for several $\mathbf{v}_j$ during the course of the iterative method for solving the correction equation $\widetilde{\mathbf{A}}\mathbf{z} = -\mathbf{r}$, this can be re-organized for efficiency by computing $\mathbf{t}_x$ once and for all and then re-using it for each of the $\mathbf{v}_j$.

```
Solve $\mathbf{K}\mathbf{t}_x = \hat{\mathbf{x}}$;
**for** j = 1,2, ..., **until** convergence,
    Produce $\mathbf{v}_j$ from the iterative method;
    $\mathbf{y} \leftarrow (\mathbf{A} - \mu\mathbf{I})\mathbf{v}_j$;
    Solve $\mathbf{K}\mathbf{t}_y = \mathbf{y}$;
    Set $\delta = \frac{\hat{\mathbf{x}}^*\mathbf{t}_y}{\hat{\mathbf{x}}^*\mathbf{t}_x}$;
    Set $\mathbf{w} = \mathbf{t}_y - \mathbf{t}_x\delta$.
```

With this scheme we only need to solve a linear system involving $\mathbf{K}$ once per step of the Krylov iteration for solving $\widetilde{\mathbf{A}}\mathbf{z} = -\mathbf{r}$. Additional details on this implementation are specified in Sleijpen and van der Vorst (1995).

With respect to parallel computation, the Jacobi–Davidson method has the same computational structure as a Krylov method. Successful parallel implementation largely depends on how well an effective preconditioner can be parallelized. An additional complication is that, even if a good preconditioner $\mathbf{K}$ exists for $\mathbf{A}$, there is no assurance that $\mathbf{K} - \theta\mathbf{I}$ will be a good one for $\mathbf{A} - \theta\mathbf{I}$. Moreover, since this operator is usually indefinite, there is often difficulty with incomplete factorization preconditioners.

6.3. JDQR: restart and deflation

The Jacobi–Davidson method can be extended to find more than one eigenpair by using *deflation* techniques. As eigenvectors converge, the iteration is continued in a subspace forced to be orthogonal to the converged eigenvectors.

Such an extension is developed in Fokkema, Sleijpen and van der Vorst (1996) to obtain an algorithm called JDQR, for computing several eigenpairs at once. The algorithm is based on the computation of a partial Schur form of $\mathbf{A}$,

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{R}_k,$$

where $\mathbf{Q}_k$ is an $(n \times k)$ orthonormal matrix, and $\mathbf{R}_k$ is a $(k \times k)$ upper triangular matrix, with $k \ll n$. As noted previously, if $(\mathbf{y}, \lambda)$ is an eigenpair of $\mathbf{R}_k$, then $(\mathbf{Q}_k\mathbf{y}, \lambda)$ is an eigenpair of $\mathbf{A}$.

To develop this algorithm, we need to derive conditions required of a new column $\mathbf{q}$ in order to expand an existing decomposition with $\mathbf{q}$ to obtain an updated partial Schur decomposition:

$$\mathbf{A}\,[\mathbf{Q}_k, \mathbf{q}] = [\mathbf{Q}_k, \mathbf{q}] \begin{bmatrix} \mathbf{R}_k & \mathbf{s} \\ 0 & \lambda \end{bmatrix}$$

with $\mathbf{Q}^*\mathbf{q} = 0$.

Equating the last column on both sides gives

$$\mathbf{A}\mathbf{q} = \mathbf{Q}_k\mathbf{s} + \mathbf{q}\lambda.$$

Multiplying this equation on the left by $\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*$ and enforcing the requirement $\mathbf{Q}_k^*\mathbf{q} = 0$ gives

$$(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*)\mathbf{A}\mathbf{q} = (\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*)(\mathbf{Q}_k\mathbf{s} + \mathbf{q}\lambda) = \mathbf{q}\lambda.$$

Finally, we arrive at

$$(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*)\mathbf{A}(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*)\mathbf{q} = \mathbf{q}\lambda,$$

which shows that the new pair $(\mathbf{q}, \lambda)$ must be an eigenpair of

$$\widetilde{\mathbf{A}} = (\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*)\mathbf{A}(\mathbf{I} - \mathbf{Q}_k\mathbf{Q}_k^*).$$

Now, we are prepared to apply the JD method so that the partial Schur decomposition may be updated.

The JDQR iteration. Assume we have $\mathbf{A}\mathbf{Q}_k = \mathbf{Q}_k\mathbf{R}_k$. Apply the JD iteration to $\widetilde{\mathbf{A}}$ and construct an orthonormal subspace basis $\mathbf{V}_\ell := [\mathbf{v}_1, \ldots, \mathbf{v}_\ell]$. Then a projected $\ell \times \ell$ matrix $\mathbf{M} = \mathbf{V}_\ell^*\widetilde{\mathbf{A}}\mathbf{V}_\ell$ is formed. We then compute the complete Schur form $\mathbf{M}\mathbf{U} = \mathbf{U}\mathbf{S}$, with $\mathbf{U}^*\mathbf{U} = \mathbf{I}$, and $\mathbf{S}$ upper triangular. This can be done with the standard QR algorithm (Golub and Van Loan 1996).

Next, $\mathbf{S}$ is reordered (using Givens' similarity transformations) to remain upper triangular but with $|\mathbf{S}_{i,i} - \tau|$ now nondecreasing with i. The first few diagonal elements of $\mathbf{S}$ then represent the eigen-approximations closest to τ, and the first few of the correspondingly reordered columns of $\mathbf{V}_k$ represent the subspace of best eigenvector approximations. If memory is limited then this subset can be used for restart. The other columns are simply discarded. The remaining subspace is expanded according to the Jacobi–Davidson method. This is repeated until sufficiently accurate Ritz values and vectors have been obtained.

After convergence of this procedure, we have $(\mathbf{q}, \lambda)$ with $\widetilde{\mathbf{A}}\mathbf{q} = \mathbf{q}\lambda$. Since $\mathbf{Q}_k^*\widetilde{\mathbf{A}} = 0$, we have $\mathbf{Q}_k^*\mathbf{q} = 0$ automatically. Now, set $\mathbf{s} = \mathbf{Q}_k^*\mathbf{A}\mathbf{q}$ to obtain

$$\mathbf{A}\mathbf{q} = \mathbf{Q}_k\mathbf{s} + \mathbf{q}\lambda,$$

and update

$$\mathbf{Q}_{k+1} := [\mathbf{Q}_k, \mathbf{q}] \quad \text{and} \quad \mathbf{R}_{k+1} := \begin{bmatrix} \mathbf{R}_k & \mathbf{s} \\ 0 & \lambda \end{bmatrix}$$

to obtain a new partial Schur decomposition of dimension $k+1$.

This process is repeated until the desired number of eigenvalues has been obtained.

7. The generalized eigenproblem

In many applications, the generalized eigenproblem $\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}\lambda$ arises naturally. A typical setting is a finite element discretization of a continuous problem where the matrix $\mathbf{B}$ arises from inner products of basis functions. In this case, $\mathbf{B}$ is symmetric and positive (semi-) definite, and for some algorithms this property is a necessary condition. Generally, algorithms are based on transforming the generalized problem to a standard problem. The details of how this is done are clearly important to efficiency and robustness. However, the fundamentals and performance of the algorithms for the standard problem carry over directly to the generalized case.

7.1. Krylov methods with spectral transformations

Perhaps the most successful general scheme for converting the generalized problem to a standard problem that is amenable to a Krylov or a subspace iteration method is to use the *spectral transformation* suggested by Ericsson and Ruhe (1980):

$$(\mathbf{A} - \sigma\mathbf{B})^{-1}\mathbf{B}x = \mathbf{x}\nu. \tag{7.1}$$

An eigenvector $\mathbf{x}$ of this transformed problem is also an eigenvector of the original problem $\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}\lambda$, with the corresponding eigenvalue given by $\lambda = \sigma + \frac{1}{\nu}$. With this transformation there is generally rapid convergence to eigenvalues near the shift σ because they are transformed to extremal well-separated eigenvalues. Perhaps an even more influential aspect of this transformation is that eigenvalues far from σ are damped (mapped near zero). It is often the case in applications that the discrete operator has eigenvalues that are large in magnitude but nonphysical and uninteresting with respect to the computation. The spectral transformation automatically overcomes the effect of these. A typical strategy is to choose σ to be a point in the complex plane that is near eigenvalues of interest and then compute the eigenvalues ν of largest magnitude in equation (7.1). It is not necessary to have σ really close to an eigenvalue. This transformation together with the implicit restarting technique is usually adequate for computing a significant number of eigenvalues near σ.

It is important to note that, even when $\mathbf{B} = \mathbf{I}$, we must generally use the shift-invert spectral transformation to find interior eigenvalues. The extreme eigenvalues of the transformed operator $\mathbf{A}_\sigma$ are generally large and well separated from the rest of the spectrum. The eigenvalues ν of largest magnitude will transform back to eigenvalues λ of the original $\mathbf{A}$ that are in a disk about the point σ. This is illustrated in Figure 7, where the $+$ symbols are the eigenvalues of $\mathbf{A}$, and the circled ones are the computed eigenvalues in the disk (dashed circle) centred at the point σ.

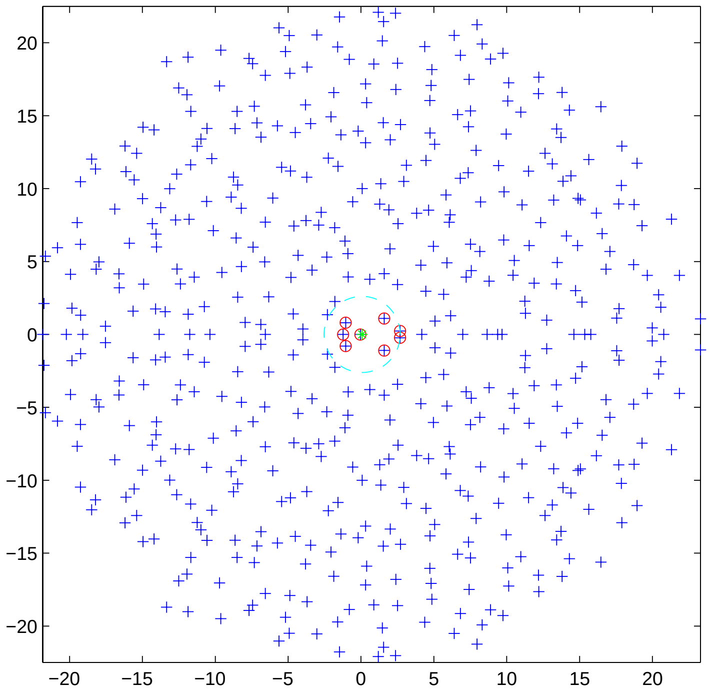

Figure 7. Eigenvalues from shift-invert

The Arnoldi process may be applied to the matrix $\mathbf{A}_\sigma := (\mathbf{A} - \sigma\mathbf{B})^{-1}\mathbf{B}$. Whenever a matrix-vector product $\mathbf{w} \leftarrow \mathbf{A}_\sigma\mathbf{v}$ is required, the following steps are performed:

$\mathbf{z} = \mathbf{B}\mathbf{v}$;
solve $(\mathbf{A} - \sigma\mathbf{B})\mathbf{w} = \mathbf{z}$ for $\mathbf{w}$.

The matrix $\mathbf{A} - \sigma\mathbf{B}$ is factored initially with a sparse direct LU-decomposition or in a symmetric indefinite factorization, and this single factorization is used repeatedly to apply the matrix operator $\mathbf{A}_\sigma$ as required.

When $\mathbf{A}$ and $\mathbf{B}$ are both symmetric and $\mathbf{B}$ is positive (semi-) definite, this approach needs to be modified slightly to preserve symmetry. In this case we can use a weighted $\mathbf{B}$ (semi-) inner product in the Lanczos/Arnoldi process (Ericsson and Ruhe 1980, Grimes *et al.* 1994, Meerbergen and Spence 1997). This amounts to replacing the computation of $\mathbf{h} \leftarrow \mathbf{V}_{j+1}^*\mathbf{w}$; and $\beta_j = \|\mathbf{f}_j\|$ with $\mathbf{h} \leftarrow \mathbf{V}_{j+1}^*\mathbf{B}\mathbf{w}$; and

$$\beta_j = \sqrt{\mathbf{f}_j^*\mathbf{B}\mathbf{f}_j},$$

respectively, in the Arnoldi process shown in Algorithm 2.

When $\mathbf{A}$ is symmetric and $\mathbf{B}$ is symmetric positive (semi-) definite, the matrix operator $\mathbf{A}_\sigma$ is self-adjoint with respect to this (semi-) inner product, that is, $\langle \mathbf{A}_\sigma\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{A}_\sigma\mathbf{y}\rangle$ for all vectors $\mathbf{x}, \mathbf{y}$, where $\langle \mathbf{w}, \mathbf{v}\rangle := \sqrt{\mathbf{w}^*\mathbf{B}\mathbf{v}}$. This implies that the projected Hessenberg matrix $\mathbf{H}$ is actually symmetric and tridiagonal and the standard three-term Lanczos recurrence is recovered with this inner product.

There is a subtle aspect to this approach when $\mathbf{B}$ is singular. The most pathological case is when $\mathrm{Null}(\mathbf{A}) \cap \mathrm{Null}(\mathbf{B}) \neq \{0\}$. If $x \in \mathrm{Null}(\mathbf{A}) \cap \mathrm{Null}(\mathbf{B})$ is nonzero, then

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}\lambda$$

for every complex number λ. This case is not treated here. A challenging but far less devastating situation occurs when this intersection is just the zero vector. In this case, $\mathrm{Null}(\mathbf{A}_\sigma) = \mathrm{Null}(\mathbf{B})$ for any σ that is not a generalized eigenvalue of the pair $(\mathbf{A}, \mathbf{B})$. Unfortunately, any nonzero vector $x \in \mathrm{Null}(\mathbf{B})$ corresponds to an *infinite eigenvalue*, since any such $\mathbf{x}$ will be an eigenvector of $\mathbf{A}_\sigma$ corresponding to the eigenvalue $\nu = 0$, and the formula $\lambda = \sigma + \frac{1}{\nu}$ indicates that $\mathbf{x}$ must correspond to an infinite eigenvalue of the original problem. Using the $\mathbf{B}$ inner product in the shift-invert Arnoldi process and requesting the eigenvalues ν of largest magnitude for $\mathbf{A}_\sigma$ through implicit restarting avoids these troublesome eigenvalues. In theory (*i.e.*, in exact arithmetic), if the starting vector $\mathbf{v}_1 = \mathbf{A}_\sigma\mathbf{v}$ is in $\mathrm{Range}(\mathbf{A}_\sigma)$ then the method cannot converge to a zero eigenvalue of $\mathbf{A}_\sigma$. However, eigenvectors are only computed approximately and these may have components in directions corresponding to infinite eigenvalues. Such components can be *purged* from a computed eigenvector $\mathbf{x}$ by replacing it with $\mathbf{x} \leftarrow \mathbf{A}_\sigma\mathbf{x}$ and renormalizing. Therefore, the recommendation is to begin the Arnoldi process with a starting vector that has been multiplied by $\mathbf{A}_\sigma$ and, after convergence, to perform a purging step on the converged approximate eigenvectors.

A clever way to perform this operation has been suggested by Ericsson and Ruhe (1980). If $\mathbf{x} = \mathbf{V}\mathbf{y}$ with $\mathbf{H}\mathbf{y} = \mathbf{y}\theta$, then

$$\mathbf{A}_\sigma \mathbf{x} = \mathbf{V}\mathbf{H}\mathbf{y} + \mathbf{f}\mathbf{e}_k^T\mathbf{y} = \mathbf{x}\theta + \mathbf{f}\mathbf{e}_k^T\mathbf{y}.$$

Replacing the $\mathbf{x}$ with the improved eigenvector approximation $\mathbf{x} \leftarrow (\mathbf{x}\theta + \mathbf{f}\mathbf{e}_k^T\mathbf{y})$ and renormalizing has the effect of purging the undesirable components without requiring any additional matrix-vector products with $\mathbf{A}_\sigma$. The residual error of the computed Ritz vector with respect to the original problem is

$$\|\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{x}\lambda\| = \|\mathbf{B}\mathbf{f}\| \frac{|\mathbf{e}_k^T\mathbf{y}|}{|\theta|^2}, \tag{7.2}$$

where $\lambda = \sigma + 1/\theta$. Since $|\theta|$ is usually quite large under the spectral transformation, this new residual is generally considerably smaller than the original.

7.2. Additional methods and accelerations

When a sparse direct factorization is possible, the shift-invert spectral transformation combined with implicitly restarted Arnoldi is probably the method of choice. However, this may not be practical in many applications. In a parallel computing environment, success of this approach also depends critically on how well the solution process for the shift-invert equations can be parallelized. Finally, if applying $\mathbf{A}_\sigma$ is very cheap then one may wish to avoid the expense of implicit restarting and complete orthogonalization.

One approach that avoids the need to keep a complete set of basis vectors is the *Bi-Lanczos method*. This biorthogonal dual-basis approach is based on a three-term recurrence that results in a nonsymmetric tridiagonal projected matrix instead of an upper Hessenberg projection. This Bi-Lanczos method is related to the QMR and Bi-CG methods for linear systems. Both methods lead to the same projected tridiagonal matrix. Freund and Nachtigal (1991) and Cullum and Willoughby (1986) have published codes for the computation of eigenvalues using this approach. However, the accuracy of these methods is a point of concern, since the projections are oblique rather than orthogonal as they are in the Arnoldi process. Also, there is no particular advantage in having the projected matrix in nonsymmetric tridiagonal form, since the only algorithms that can take advantage of the structure are generally based on hyperbolic rotations, and are of questionable numerical stability.

An alternative spectral transformation that can be effective in linear stability analysis in CFD problems is the generalized Cayley transformation

$$\mathbf{C} := (\mathbf{A} - \sigma\mathbf{B})^{-1}(\mathbf{A} - \lambda\mathbf{B}).$$

An important aspect of this transformation is the additional control on the image of the left half plane (say) under the transformation. A detailed study may be found in Garratt (1991) and Meerbergen and Spence (1997). Lehoucq and Salinger (2001) make particularly effective use of this transformation in a stability analysis of a simulation of a CVD reactor with over four million variables. More recently, 16 million variable problems of this type have been solved (Burroughs, Romero, Lehoucq and Salinger 2001).

The use of inexact forms of the Cayley transform is studied in Meerbergen (1996), where the required inverse operation is approximated by a few steps of an iterative method, for Arnoldi's method. The wanted eigensolutions are solutions of $\mathbf{Cx} = 0$. The essential part $\mathbf{A} - \lambda\mathbf{B}$ is computed exactly and the inexact inversion of $\mathbf{A} - \sigma\mathbf{B}$ may be viewed as a kind of preconditioning. Indeed, this technique has a close relationship to polynomial preconditioning. The inexact Cayley transform is well suited to parallel computing since the dominant computational elements are matrix-vector products instead of direct linear solves.

Ruhe (1994*b*) introduced a remarkable generalization of the Krylov space that admits the application of several different shift-invert transforms within the same iteration. This is called *rational Krylov subspace* (RKS) iteration and the transformations are of the form

$$(\delta_j \mathbf{A} - \gamma_j \mathbf{B})^{-1}(\sigma_j \mathbf{A} - \rho_j \mathbf{B}),$$

in which the coefficients may be different for each iteration step j. With respect to the subspace with these operators, the given problem is projected onto a small generalized system

$$(\zeta \mathbf{K}_{j,j} - \eta \mathbf{L}_{j,j})\mathbf{y} = 0,$$

where $\mathbf{K}_{j,j}$ and $\mathbf{L}_{j,j}$ are upper Hessenberg matrices of dimension j. This small system may be solved by the QZ algorithm in order to obtain approximate values for an eigenpair. The parameters in RKS can be chosen to obtain faster convergence to interior eigenvalues. When combined with a certain deflation scheme, a considerable number of eigenvalues can be computed without being forced to construct a large basis set. Eigenvectors can be written to auxiliary storage as needed. For a comparison of RKS and Arnoldi, see Ruhe (1994*a*, 1994*b*). Again, successful parallelization for this approach depends on how well linear systems with the matrix $\delta_j \mathbf{A} - \gamma_j \mathbf{B}$ can be solved to sufficiently high accuracy.

Clearly, the most straightforward alternative to solving the shift-invert equations directly is to use a preconditioned iterative method to solve them approximately. However, there are several difficulties. The shifted matrix is often ill-conditioned because σ will be chosen near an interesting eigenvalue. Moreover, this shifted matrix will usually be indefinite (or have indefinite symmetric part). These are the conditions that are most difficult for iterative solution of linear systems. A further difficulty is that each linear system must be solved to a greater accuracy than the desired accuracy of the eigenvalue calculation. Otherwise, each step of the Lanczos/Arnoldi process will essentially involve a different matrix operator. The approach can be quite successful, however, if done with care. A good example of this may be found in Lehoucq and Salinger (2001).

Subspace iteration methods are more amenable to admitting inaccurate approximate solutions to the shift-invert equations. This has already been discussed in Section 6.2. The Jacobi–Davidson approach can be adapted nicely to the generalized problem and is particularly well suited to the introduction of inaccurate approximate solutions.

For a good overview of subspace iteration methods, see Saad (1992). There are several other methods that allow the possibility of inexact solves and preconditioning in eigenvalue computations. Two of these are the LOBPCG method developed in Knyazev (2001) and the TRQ method developed in Sorensen and Yang (1998).

Knyazev (2001) presents numerical evidence to suggest that LOBPCG performs for symmetric positive definite eigenproblems as the preconditioned conjugate gradient method performs for symmetric positive definite linear systems. In Sorensen and Yang (1998), the TRQ method is derived as a truncation of the RQ iteration. This is just like the QR method with the exception that the shifted matrices are factored into an orthogonal $\mathbf{Q}$ times an upper triangular $\mathbf{R}$. Quadratic convergence takes place in the leading column of $\mathbf{Q}$ and preconditioned inexact solves are possible to complete the update equations. This scheme is very closely related to the JDQR method.

7.3. The Jacobi–Davidson QZ algorithm

The Jacobi–Davidson method can be modified for the generalized eigenproblem without having to transform the given problem to a standard eigenproblem. In this formulation, called JDQZ (Fokkema *et al.* 1996), explicit inversion of matrices is not required. The method is developed with orthogonal projections and the theme is once again to compute a partial (generalized) Schur decomposition. A subspace is generated onto which the given eigenproblem is projected. The much smaller projected eigenproblem is solved by standard direct methods, and this leads to approximations for the eigensolutions of the given large problem. Then, a correction equation for a selected eigenpair is set up. The solution of the correction equation defines an orthogonal correction for the current eigenvector approximation. The correction, or an approximation for it, is used for the expansion of the subspace and the procedure is repeated.

The subspace projection leads to a formulation that may be viewed as an inexact truncated form of the QZ factorization. The algorithm is designed to compute a few eigenvalues of $\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}\lambda$ close to a given target $\tau \in \mathbb{C}$. Given a low-dimensional subspace $\mathcal{S} = \text{Range}(\mathbf{V}_k)$, eigenvector approximations called Petrov–Ritz values are extracted from a small projected problem obtained through a Petrov–Galerkin projection.

Here $\mathbf{V}_k$ is an $(n \times k)$ matrix with orthonormal columns $\mathbf{v}_j$. A Petrov–Galerkin condition is to define a Petrov–Ritz pair $(\mathbf{x}, \theta)$, where $\mathbf{x} \in \mathcal{S}$. We require

$$\langle \mathbf{w}, \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{x}\theta \rangle = 0, \quad \text{for all } \ \mathbf{w} \in \text{Range}(\mathbf{W}_k),$$

where $\mathbf{W}_k$ is another $(n \times k)$ matrix with orthonormal columns $\mathbf{w}_j$. This gives a small projected problem of order k:

$$\mathbf{W}_k^*\mathbf{A}\mathbf{V}_k\mathbf{y} - \mathbf{W}_k^*\mathbf{B}\mathbf{V}_k\mathbf{y}\theta = 0. \tag{7.3}$$

For each eigenpair $(\mathbf{y}, \theta)$, we obtain a Petrov–Ritz vector $\mathbf{x} = \mathbf{V}_k\mathbf{y}$ and Petrov–Ritz value θ as approximate eigenpairs for the original problem.

Using essentially the same approach described for the standard problem (see Section 6.3), the basis sets $\mathbf{V}_k$ and $\mathbf{W}_k$ are each increased one dimension by including directions obtained from a residual correction equation. The method is briefly described here. For full details one should consult Fokkema *et al.* (1996).

First the QZ algorithm (Golub and Van Loan 1996) is used to reduce (7.3) to a generalized Schur form. This provides orthogonal $(k \times k)$ matrices $\mathbf{U}_R$ and $\mathbf{U}_L$, and upper triangular $(k \times k)$ matrices $\mathbf{S}_A$ and $\mathbf{S}_B$, such that

$$\mathbf{U}_L^* \left(\mathbf{W}_k^*\mathbf{A}\mathbf{V}_k\right) \mathbf{U}_R \; = \; \mathbf{S}_A, \tag{7.4}$$

$$\mathbf{U}_L^* \left(\mathbf{W}_k^*\mathbf{B}\mathbf{V}_k\right) \mathbf{U}_R \; = \; \mathbf{S}_B. \tag{7.5}$$

The decomposition is ordered (by similarity transformations) so that the leading diagonal elements of $\mathbf{S}_A$ and $\mathbf{S}_B$ represent the eigenvalue approximation closest to the target value τ. The approximation for the eigenpair is then taken as

$$(\tilde{\mathbf{q}}, \theta) := (\mathbf{V}_k\mathbf{U}_R(:,1), \mathbf{S}_B(1,1)/\mathbf{S}_A(1,1)), \tag{7.6}$$

assuming that $\mathbf{S}_A(1,1) \neq 0$. This gives a *residual vector*:

$$\mathbf{r} := \mathbf{A}\tilde{\mathbf{q}} - \mathbf{B}\tilde{\mathbf{q}}\theta.$$

To obtain a correction equation analogous to (6.4), we define an auxiliary vector $\tilde{\mathbf{w}}\gamma = \mathbf{A}\tilde{\mathbf{q}} - \mathbf{B}\tilde{\mathbf{q}}\tau$, where γ is such that $||\tilde{\mathbf{w}}||_2 = 1$. Then a correction equation is defined to provide a correction $\mathbf{z} \perp \tilde{\mathbf{q}}$:

$$(\mathbf{I} - \tilde{\mathbf{w}}\tilde{\mathbf{w}}^*)(\mathbf{A} - \theta\mathbf{B})(\mathbf{I} - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^*)\mathbf{z} = -\mathbf{r}. \tag{7.7}$$

In practice, only a few steps of a preconditioned iterative method are done to get an approximate solution to (7.7).

The approximation for $\mathbf{z}$ is then further orthogonalized with respect to $\mathbf{V}_k$ to get the new basis vector $\mathbf{v}_{k+1}$ in the direction of $(\mathbf{I} - \mathbf{V}_k\mathbf{V}_k^*)\mathbf{z}$. The expansion vector $\mathbf{w}_{k+1}$ is taken in the direction $(\mathbf{I} - \mathbf{W}_k\mathbf{W}_k^*)(\mathbf{A}\mathbf{z} - \mathbf{B}\mathbf{z}\tau)$. This gives a brief description of the *harmonic Petrov value approach* proposed in Fokkema *et al.* (1996).

7.4. JDQZ: restart and deflation

As in JDQR, deflation and restarting must be employed to find several eigenvalues and vectors simultaneously. As eigenvectors converge, the iteration is continued in a subspace orthogonal to the converged vectors. The algorithm is based on the computation of a partial generalized Schur form for the matrix pair $(\mathbf{A}, \mathbf{B})$:

$$\mathbf{A}\mathbf{Q}_k = \mathbf{Z}_k\mathbf{S}_k \quad \text{and} \quad \mathbf{B}\mathbf{Q}_k = \mathbf{Z}_k\mathbf{T}_k,$$

in which $\mathbf{Q}_k$ and $\mathbf{Z}_k$ are $(n \times k)$ orthonormal matrices and $\mathbf{S}_k$, $\mathbf{T}_k$ are $(k \times k)$ upper triangular matrices, with $k \ll n$. The scheme is more complicated but essentially follows the ideas described previously for JDQR. The full details may be found in Fokkema *et al.* (1996).

8. Eigenvalue software

Several software packages were developed during the 1980s for large-scale symmetric problems. Perhaps the most influential of these was Grimes *et al.* (1994). This block Lanczos code has been a mainstay of structural analysis calculations in industrial applications. It has been updated many times and is still the most heavily used code in this field. Considerable progress has been made over the past decade on the production of high-quality mathematical software for large nonsymmetric eigenvalue problems. Many packages are freely available online, and may be found via netlib.

A few of these are:

Lanczos (`http://www.netlib.org/`)

Authors: Jane Cullum and Ralph A. Willoughby
Description: Lanczos Algorithms for computing a few eigenvalues and eigenvectors of a large (sparse) symmetric matrix, real symmetric and Hermitian matrices; singular values and vectors of real, rectangular matrices (Fortran)
Reference: Cullum and Willoughby (1985)

SRRIT (`http://www.netlib.org/`)

Authors: Z. Bai and G. W. Stewart
Description: Subspace iteration to calculate the dominant invariant subspace of a nonsymmetric matrix (Fortran)
Reference: Bai and Stewart (1997)

ARNCHEB (`http://www.cerfacs.fr/~chatelin/`)

Authors: T. Braconnier and F. Chatelin
Description: Arnoldi–Chebyshev restarted method for computing a few eigenvalues and vectors of large, unsymmetric sparse matrices (Fortran)
Reference: Users' Guide (`http://www.cerfacs.fr/~chatelin/`)

LOBPCG (`http://www-math.cudenver.edu/~aknyazev/software/CG`)

Author: A. Knyazev
Description: Locally optimal block preconditioned conjugate gradient method for a few eigenvalues and vectors of large symmetric (or Hermitian) matrices (Matlab)
Reference: Knyazev (2001)

Laso (`http://www.netlib.org/`)

Author: D. Scott
Description: Lanczos method for a few eigenvalues and eigenvectors of a large (sparse) symmetric matrix (Fortran)
Reference: Parlett and Scott (1979)

SVDpack (`http://www.netlib.org/`)

Authors: M. W. Berry and M. Liang
Description: Computes a partial SVD of large sparse non-Hermitian complex matrices using the Lanczos algorithm for $\mathbf{A}^*\mathbf{A}$ with selective reorthogonalization (Fortran)
Reference: Berry (1992)

IRBL (`http://www.cs.bsu.edu/~jbaglama/#Software`)

Authors: J. Baglama, D. Calvetti and L. Reichel
Description: Block implicitly restarted Lanczos with Leja points as shifts.

JDQR, JDQZ (`http://www.math.uu.nl/people/sleijpen/JD_software`)

Author: G. L. G. Sleijpen
Description: JDQR and JDQZ implementations of Jacobi–Davidson method for a partial Schur decomposition corresponding to a selected subset of eigenvalues (eigenvectors also computed on request). Symmetric, nonsymmetric, generalized problems solved (Matlab)
Reference: Sleijpen and van der Vorst (1995), Fokkema *et al.* (1996)

ARPACK (`http://www.caam.rice.edu/software/ARPACK/`)

Authors: R. B. Lehoucq, D. C. Sorensen, and C. Yang
Description: Implicitly restarted Arnoldi method for computing a partial Schur decomposition corresponding to a selected subset of eigenvalues (eigenvectors also computed on request). Symmetric, nonsymmetric, generalized and SVD problems solved (Fortran)
Reference: Lehoucq, Sorensen and Yang (1998)

We should also mention the codes available in the Harwell Subroutine Library (HSL). These are freely available to UK academics, but not in general. In particular, the code EB13 based on Scott (200x) is available for nonsymmetric problems.

8.1. Software design

Today's software designers are faced with many new options in languages, design options, and computational platforms. However, certain principles can lead to robust software that is both portable and efficient over a wide variety of computing platforms.

When designing general-purpose software for use in the public domain, it is important to adopt a development strategy that will meet the goals of robustness, efficiency, and portability. Two very important principles are modularity and independence from specific vendor-supplied communication and performance libraries.

In this final section, we discuss some design and performance features of the eigenvalue software ARPACK. This is a collection of Fortran77 subroutines based on the IRAM described in Algorithm 3. This software can solve large-scale non-Hermitian or Hermitian (standard and generalized) eigenvalue problems. It has been used on a wide range of applications. P_ARPACK is a parallel extension to the ARPACK library and is designed for distributed memory message passing systems. The message passing layers currently supported are BLACS and MPI (MPI Forum 1994, Dongarra and Whaley 1995). Performance and portability are attained simultaneously because of the modular construction of the dense linear algebra operations. These are based on the Level 2 and Level 3 BLAS (Dongarra *et al.* 1988, Dongarra, DuCroz, Duff and Hammarling 1990) for matrix-vector and matrix-matrix operations and on LAPACK (Anderson *et al.* 1992) for higher-level dense linear algebra routines.

The important features of ARPACK and P_ARPACK are as follows.

- A reverse communication interface.
- Computes k eigenvalues that satisfy a user-specified criterion such as largest real part, largest absolute value, *etc.*

- A fixed predetermined storage requirement of $n \cdot \mathcal{O}(k) + \mathcal{O}(k^2)$ bytes.
- Driver routines are included as templates for implementing various spectral transformations to enhance convergence and to solve the generalized eigenvalue problem, or the SVD problem.
- Special consideration is given to the generalized problem $\mathbf{Ax} = \mathbf{Bx}\lambda$ for singular or ill-conditioned symmetric positive semi-definite $\mathbf{B}$.
- A Schur basis of dimension k that is numerically orthogonal to working precision is always computed. These are also eigenvectors in the Hermitian case. In the non-Hermitian case eigenvectors are available on request. Eigenvalues are computed to a user-specified accuracy.

Reverse communication
Reverse communication is an artifact of certain restrictions in the Fortran language; with reverse communication, control is returned to the calling program when interaction with the matrix is required. (For the C++ programmer, reverse communication is the Fortran substitute for defining functions specific to the class of matrices.) This is a convenient interface for experienced users. However, it seems to be a difficult concept to grasp for inexperienced users. Even though it is extremely useful for interfacing with large application codes, the software maintenance problems imposed on the developers are very demanding.

This interface avoids having to express a matrix-vector product through a subroutine with a fixed calling sequence. This means that the user is free to choose any convenient data structure for the matrix representation. Also, it is up to the user to partition the matrix-vector product in the most favourable way for parallel efficiency. Moreover, if the matrix is not available explicitly, the user is free to express the action of the matrix on a vector through a subroutine call or a code segment. It is not necessary to conform to a fixed format for a subroutine interface, and hence there is no need to communicate data through the use of `COMMON`.

A typical use of this interface is illustrated as follows:

```
10  continue
    call snaupd (ido, bmat, n, which,...,workd,..., info)
    if (ido .eq. newprod) then
       call matvec ('A', n, workd(ipntr(1)), workd(ipntr(2)))
    else
       return
    endif
    go to 10
```

This shows a code segment of the routine the user must write to set up the reverse communication call to the top level ARPACK routine `snaupd` to solve a nonsymmetric eigenvalue problem. The action requested of the calling program is specified by the reverse communication parameter `ido`. In this case the requested action is multiply the vector held in the array `workd` beginning at location `ipntr(1)` and and then to insert into the array `workd` beginning at location `ipntr(2)`. Here a call is made to a subroutine `matvec`. However, it is only necessary to supply the action of the matrix on the specified vector and put the result in the designated location. Because of this, reverse communication is very flexible and even provides a convenient way to use ARPACK interfaced with code written in another language such as C or C++.

8.2. Parallel aspects

The parallelization paradigm found to be most effective for ARPACK on distributed memory machines was to provide the user with a *single program multiple data* (SPMD) template. This means there are many copies of the same program running on multiple processors executing the same instruction streams on different data. The parallelization scheme described here is well suited to all of the methods discussed earlier, since they all share the basic needs of orthogonalizing a new vector with respect to a current basis for a subspace. They also share the need to apply a linear operator to a vector.

The reverse communication interface provides a means for a very simple SPMD parallelization strategy. Reverse communication allows the P_ARPACK codes to be parallelized internally without imposing a fixed parallel decomposition on the matrix or the user-supplied matrix-vector product. Memory and communication management for the matrix-vector product $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}$ can be optimized independently of P_ARPACK. This feature enables the use of various matrix storage formats as well as calculation of the matrix elements as needed.

The calling sequence to ARPACK remains unchanged except for the addition of an MPI communicator (MPI Forum 1994, Dongarra and Whaley 1995). Inclusion of the communicator is necessary for global communication as well as managing I/O.

The numerically stable generation of the Arnoldi factorization

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k + \mathbf{f}_k\mathbf{e}_k^T$$

coupled with an implicit restarting mechanism is the basis of the ARPACK codes. The simple parallelization scheme used for P_ARPACK is as follows:

- $\mathbf{H}_k$ replicated on every processor
- $\mathbf{V}_k$ is distributed across a 1D processor grid (blocked by rows)
- $\mathbf{f}_k$ and workspace distributed accordingly.

The SPMD code looks essentially like the serial code except that the local block of the set of Arnoldi vectors, $\mathbf{V}_{\mathrm{loc}}$, is passed in place of $\mathbf{V}$, and n_{loc}, the dimension of the local block, is passed instead of n.

With this approach there are only two communication points within the construction of the Arnoldi factorization inside P_ARPACK: computation of the 2-norm of the distributed vector $\mathbf{f}_k$ and the orthogonalization of $\mathbf{f}_k$ to $\mathbf{V}_k$ using classical Gram–Schmidt with DGKS correction (Daniel, Gragg, Kaufman and Stewart 1976). Additional communication will typically occur in the user-supplied matrix-vector product operation as well. Ideally, this product will only require nearest neighbour communication among the processes. Typically, the blocking of $\mathbf{V}$ coincides with the parallel decomposition of the matrix $\mathbf{A}$. The user is free to select an appropriate blocking of $\mathbf{V}$ to achieve optimal balance between the parallel performance of P_ARPACK and the user-supplied matrix-vector product.

The SPMD parallel code looks very similar to that of the serial code. Assuming a parallel version of the subroutine `matvec`, an example of the application of the distributed interface is illustrated as follows:

```
10  continue
    call psnaupd (comm, ido, bmat, nloc, which, ...,
   *                      Vloc , ... lworkl, info)
    if (ido .eq. newprod) then
       call matvec ('A', nloc, workd(ipntr(1)), workd(ipntr(2)))
    else
       return
    endif
    go to 10
```

Here, `nloc` is the number of rows in the block `Vloc` of $\mathbf{V}$ that has been assigned to this node process.

The blocking of $\mathbf{V}$ is generally determined by the parallel decomposition of the matrix A. For parallel efficiency, this blocking must respect the configuration of the distributed memory and interconnection network. Logically, the $\mathbf{V}$ matrix is partitioned by blocks

$$\mathbf{V}^T = (\mathbf{V}^{(1)^T}, \mathbf{V}^{(2)^T}, \dots, \mathbf{V}^{(\mathrm{nproc})^T}),$$

with one block per processor and with $\mathbf{H}$ replicated on each processor. The explicit steps of the CGS process taking place on the jth processor are shown in Algorithm 8.

Note that the function `gnorm` at step (1) is meant to represent the global reduction operation of computing the norm of the distributed vector $\mathbf{f}_k$ from

```
(1)  $\beta_k \leftarrow \text{gnorm}(\|\mathbf{f}_k^{(*)}\|);\;\; \mathbf{v}_{k+1}^{(j)} \leftarrow \mathbf{f}_k^{(j)} \cdot \frac{1}{\beta_k};$
(2)  $\mathbf{w}^{(j)} \leftarrow (\mathbf{A}_{\text{loc}}) v_{k+1}^{(j)};$
(3)  $\begin{pmatrix} \mathbf{h} \\ \alpha \end{pmatrix}^{(j)} \leftarrow \begin{pmatrix} \mathbf{V}_k^{(j)T} \\ \mathbf{v}_{k+1}^{(j)T} \end{pmatrix} \mathbf{w}^{(j)};\; \begin{pmatrix} \mathbf{h} \\ \alpha \end{pmatrix} \leftarrow \text{gsum}\left[\begin{pmatrix} \mathbf{h} \\ \alpha \end{pmatrix}^{(*)}\right]$
(4)  $\mathbf{f}_{k+1}^{(j)} \leftarrow \mathbf{w}^{(j)} - (\mathbf{V}_k, \mathbf{v}_{k+1})^{(j)} \begin{pmatrix} \mathbf{h} \\ \alpha \end{pmatrix};$
(5)  $\mathbf{H}_{k+1} \leftarrow \begin{pmatrix} \mathbf{H}_k & \mathbf{h} \\ \beta_k & \mathbf{e}_k^T \end{pmatrix};$
(6)  $\mathbf{V}_{k+1}^{(j)} \leftarrow (\mathbf{V}_k, \mathbf{v}_{k+1})^{(j)};$
```

Algorithm 8. The explicit steps of the process responsible for the j block

the norms of the local segments $\mathbf{f}_k^{(j)}$, and the function `gsum` at step (3) is meant to represent the global sum of the local vectors $\mathbf{h}^{(j)}$ so that the quantity $\mathbf{h} = \sum_{j=1}^{\text{nproc}} \mathbf{h}^{(j)}$ is available to each process on completion. These are the only two communication points within this algorithm. The remainder is perfectly parallel. Additional communication will typically occur at step (2). Here the operation $(\mathbf{A}_{\text{loc}})\mathbf{v}$ is meant to indicate that the user-supplied matrix-vector product is able to compute the local segment of the matrix-vector product $\mathbf{Av}$ that is consistent with the partition of $\mathbf{V}$. Ideally, this would only involve nearest neighbour communication among the processes.

Since $\mathbf{H}$ is replicated on each processor, the implicit restart mechanism described in Section 4.4 remains untouched. The only difference is that the local block $\mathbf{V}^{(j)}$ is in place of the full matrix $\mathbf{V}$. Operations associated with implicit restarting are perfectly parallel with this strategy.

All operations on the matrix $\mathbf{H}$ are replicated on each processor. Thus there are no communication overheads. However, the replication of $\mathbf{H}$ and the shift selection and application to $\mathbf{H}$ on each processor amount to a serial bottleneck limiting the scalability of this scheme when k grows with n. Nevertheless, if k is fixed as n increases then this scheme scales linearly with n, as we shall demonstrate with some computational results. In the actual implementation, separate storage is not required for the Q_i. Instead, it is represented as a product of 2×2 Givens or 3×3 Householder transformations that are applied directly to update Q. On completion of this accumulation of Q, the operation $\mathbf{V}_m^{(j)} \leftarrow \mathbf{V}_m^{(j)}\mathbf{Q}$ occurs independently on each processor j using the Level 3 BLAS operation _GEMM.

An important aspect to this approach is that changes to the serial version of ARPACK were minimal. Only eight routines were affected in a minimal way. These routines either required a change in norm calculation to accommodate distributed vectors (step (1)), modification of the distributed dense matrix-vector product (step (4)), or inclusion of the context or communicator for I/O (debugging/tracing).

8.3. Communication and synchronization

On many shared memory MIMD architectures, a level of parallelization can be accomplished through compiler options alone, without requiring any modifications to the source code. For example, on the SGI Power Challenge architecture, the MIPSpro F77 compiler uses a Power Fortran Accelerator (PFA) preprocessor to uncover the parallelism in the source code automatically. PFA is an optimizing Fortran preprocessor that discovers parallelism in Fortran code and converts those programs to parallel code. A brief discussion of implementation details for ARPACK using PFA preprocessing may be found in Debicki, Jedrzejewski, Mielewski, Przybyszewski and Mrozowski (1995). The effectiveness of this preprocessing step is still dependent on how suitable the source code is for parallelization. Since most of the vector and matrix operations for ARPACK are accomplished via BLAS and LAPACK routines, access to efficient parallel versions of these libraries alone will provide a reasonable level of parallelization.

For distributed memory implementations, message passing between processes must be explicitly addressed within the source code, and numerical computations must take into account the distribution of data. In addition, for the parallel code to be portable, the communication interface used for message passing must be supported on a wide range of parallel machines and platforms. For P_ARPACK, this portability is achieved via the *basic linear algebra communication subprograms* (BLACS) (Dongarra and Whaley 1995) developed for the ScaLAPACK project and *message passing interface* (MPI) (MPI Forum 1994).

8.4. Parallel performance

P_ARPACK has been run on a wide variety of parallel processors. The simple SPMD strategy has proved to be very effective. Near-linear scalability has been demonstrated on massively parallel machines for the internal dense linear algebra operations required to implement the IRAM. However, such scalability relies entirely on the parallel efficiency of the user-supplied matrix-vector product or linear solves when shift-invert is used. A synopsis of such performance results is available in Maschhoff and Sorensen (1996).

Perhaps more important is the ability to solve real problems. A very impressive computation has been done by Lehoucq and Salinger (2001) on a linear stability analysis of a CVD reactor. The problem involved four million variables resulting from a 3D finite element model. They used P_ARPACK on the Sandia-Intel 1024 processor Teraflop machine. A Cayley transformation $(\mathbf{A} - \sigma_1 \mathbf{B})\mathbf{w} = (\mathbf{A} + \sigma_2 \mathbf{B})\mathbf{v}$ was used to accelerate convergence and to better isolate the rightmost eigenvalues. The AZTEC package for iterative solution of linear systems was used to implement this. They selected an ILUT preconditioner with GMRES. In this calculation, P_ARPACK only contributed to about 5% of the total computation time. This is typical of many applications. The application of the linear operator (in this case the Cayley-transformed matrix) usually dominates the computation. The internal operations required for IRAM are generally inconsequential when compared to the application of the linear operator.

The Lehoucq and Salinger paper reports some very impressive results on bifurcation as well as stability analysis. They also give a very interesting study of the two-step CGS orthogonalization scheme in the context of the GMRES calculations required to solve the linear systems for the Cayley transformation. This is pertinent to all of the methods discussed here and is of particular interest in the implementation of the Arnoldi factorization that underlies GMRES and also ARPACK. Two-step CGS orthogonalization is classical Gram–Schmidt followed by one step of the DGKS correction described previously. This is done at every orthogonalization step. Considerable experience with this option for CGS has demonstrated completely reliable orthogonalization properties of many orthogonalization steps. It completely resolves the numerical problems with CGS.

Lehoucq and Salinger compare the performance of CGS to that of modified Gram–Schmidt. A comparison of computational times is shown in Figure 8.

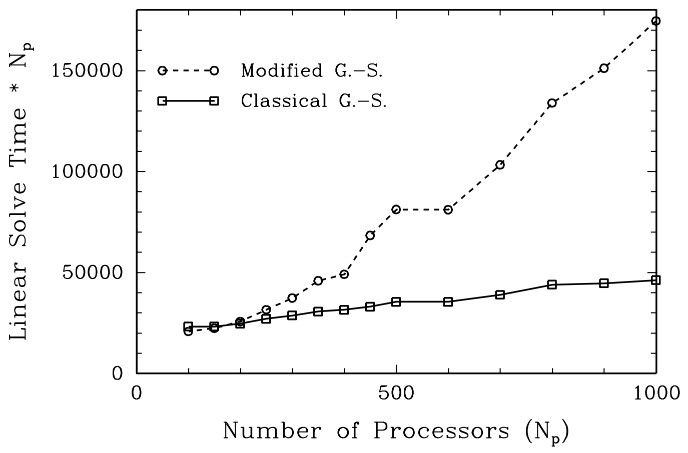

Figure 8. DGKS correction

This comparison shows that two-step CGS scales almost linearly, while MGS has very poor scalability properties. This is due to the many additional communication points needed for vector-vector operations (Level 1 BLAS) in comparison to the matrix-vector (Level 2 BLAS) formulation available with CGS. In these calculations, problem size is increased in proportion to the number of processors. Perfect scaling would give a flat horizontal graph indicating a constant computational time.

It should be noted that (unrestarted) GMRES will give the same numerical result for the linear system with either orthogonalization scheme. However, the Intel machine (called ASCI Red) has very fast communication and hence these results would be even more dramatic on most other massively parallel platforms.

8.5. Summary

The implementation of P_ARPACK is portable across a wide range of distributed memory platforms. The portability of P_ARPACK is achieved by use of the BLACS and MPI. With this strategy, it takes very little effort to port P_ARPACK to a wide variety of parallel platforms. It has been installed and successfully tested on many massively parallel systems.

9. Conclusions and acknowledgements

This introduction to the current state of the art in methods and software for large-scale eigenvalue problems has necessarily been limited. There are many excellent researchers working in the area. This discussion has focused on IRLM and JDQR methods. We have tried to include brief descriptions of most of the techniques that have been developed recently, but there are certainly unintentional omissions. The author apologizes for these.

The recent advances for nonsymmetric problems have been considerable. However, there is much left to be done. The areas of preconditioning and other forms of convergence acceleration are very challenging. The ability to compute interior eigenvalues reliably, without shift and invert spectral transformations is, at this point, out of reach.

The author owes many debts of gratitude to other researchers in this area. Several have contributed directly to this work. Of particular note are Chris Beattie, Mark Embree, Lothar Reichel, Rich Lehoucq, Chao Yang and Kristi Maschhoff. A final note of thanks goes to Arieh Iserles for his encouragement and unbelievable patience.

REFERENCES

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov and D. Sorensen (1992), *LAPACK User's Guide*, SIAM, Philadelphia, PA.

J. Baglama, D. Calvetti and L. Reichel (1996), 'Iterative methods for the computation of a few eigenvalues of a large symmetric matrix', *BIT* **36**, 400–440.

J. Baglama, D. Calvetti and L. Reichel (1998), 'Fast Leja points', *ETNA* **7**, 124–140.

Z. Bai and G. W. Stewart (1997), 'SRRIT: A FORTRAN subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix', *ACM Trans. Math. Software* **23**, 494.

C. Beattie, M. Embree and J. Rossi (2001), Convergence of restarted Krylov subspaces to invariant subspaces, Numerical Analysis Technical Report 01/21, OUCL, Oxford, UK.

M. Berry (1992), 'Large scale singular value computations', *Supercomput. Appl.* **6**, 13–49.

E. A. Burroughs, L. A. Romero, R. B. Lehoucq and A. J. Salinger (2001), Large scale eigenvalue calculations for computing the stability of buoyancy driven flows, Technical Report 2001-0113J, Sandia National Laboratories. Submitted to *J. Comput. Phys.*

D. Calvetti, L. Reichel and D. Sorensen (1994), 'An implicitly restarted Lanczos method for large symmetric eigenvalue problems', *ETNA* **2**, 1–21.

M. Crouzeix, B. Philippe and M. Sadkane (1994), 'The Davidson method', *SIAM J. Sci. Comput.* **15**, 62–76.

J. Cullum and W. E. Donath (1974), A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace for large, sparse symmetric matrices, in *Proc. 1974 IEEE Conference on Decision and Control*, New York, pp. 505–509.

J. Cullum and R. A. Willoughby (1981), 'Computing eigenvalues of very large symmetric matrices: An implementation of a Lanczos algorithm with no reorthogonalization', *J. Comput. Phys.* **434**, 329–358.

J. Cullum and R. A. Willoughby (1985), *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol. 1: Theory, Birkhäuser, Boston, MA.

J. Cullum and R. A. Willoughby (1986), A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices, in *Large Scale Eigenvalue Problems* (J. Cullum and R. A. Willoughby, eds), North-Holland, Amsterdam, pp. 193–240.

J. Daniel, W. B. Gragg, L. Kaufman and G. W. Stewart (1976), 'Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization', *Math. Comput.* **30**, 772–795.

E. R. Davidson (1975), 'The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices', *J. Comput. Phys.* **17**, 87–94.

E. R. Davidson (1993), 'Monster matrices: Their eigenvalues and eigenvectors', *Comput. Phys.* **7**, 519–522.

M. P. Debicki, P. Jedrzejewski, J. Mielewski, P. Przybyszewski and M. Mrozowski (1995), Application of the Arnoldi method to the solution of electromagnetic eigenproblems on the multiprocessor power challenge architecture, Preprint 19/95, Department of Electronics, Technical University of Gdansk, Gdansk, Poland.

J. Dongarra and R. C. Whaley (1995), A User's Guide to the BLACS v1.0, Technical Report UT CS-95-281, LAPACK Working Note #94, University of Tennessee.

J. J. Dongarra, J. DuCroz, I. Duff and S. Hammarling (1990), 'A set of Level 3 Basic Linear Algebra Subprograms: Model implementation and test programs', *ACM Trans. Math. Software* **16**, 18–28.

J. J. Dongarra, J. DuCroz, S. Hammarling and R. Hanson (1988), 'An extended set of Fortran Basic Linear Algebra Subprograms', *ACM Trans. Math. Software* **14**, 1–17.

T. Ericsson and A. Ruhe (1980), 'The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems', *Math. Comput.* **35**, 1251–1268.

B. Fischer and L. Reichel (1989), 'Newton interpolation in Chebyshev and Fejér points', *Math. Comput.* **53**, 265–278.

D. R. Fokkema, G. L. G. Sleijpen and H. A. van der Vorst (1996), Jacobi–Davidson style QR and QZ algorithms for the partial reduction of matrix pencils, Technical Report 941, Mathematical Institute, Utrecht University.

R. W. Freund (1992), 'Conjugate gradient-type methods for a linear systems with complex symmetric coefficient matrices', *SIAM J. Sci. Comput.* **13**, 425–448.

R. W. Freund and N. M. Nachtigal (1991), 'QMR: A quasi-minimal residual method for non-Hermitian linear systems', *Numer. Math.* **60**, 315–339.

D. Gaier (1987), *Lectures on Complex Approximation*, Birkhäuser.

T. J. Garratt (1991), The numerical detection of Hopf bifurcations in large systems arising in fluid mechanics, PhD thesis, University of Bath, School of Mathematical Sciences, Bath, UK.

G. H. Golub and R. Underwood (1977), The block Lanczos method for computing eigenvalues, in *Mathematical Software III* (J. Rice, ed.), Academic Press, New York, pp. 361–377.

G. H. Golub and C. F. Van Loan (1996), *Matrix Computations*, The Johns Hopkins University Press, Baltimore.

R. G. Grimes, J. G. Lewis and H. D. Simon (1994), 'A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems', *SIAM J. Matrix Anal. Appl.* **15**, 228–272.

Z. Jia (1995), 'The convergence of generalized Lanczos methods for large unsymmetric eigenproblems', *SIAM J. Matrix Anal. Appl.* **16**, 843–862.

W. Karush (1951), 'An iterative method for finding characteristics vectors of a symmetric matrix', *Pacific J. Math.* **1**, 233–248.

A. Knyazev (2001), 'Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method', *SIAM J. Sci. Comput.* **23**, 517–541.

C. Lanczos (1950), 'An iteration method for the solution of the eigenvalue problem of linear differential and integral operators', *J. Res. Nat. Bur. Standards* **45**, 255–282. Research Paper 2133.

C. Lawson, R. Hanson, D. Kincaid and F. Krogh (1979), 'Basic Linear Algebra Subprograms for Fortran usage.', *ACM Trans. Math. Software* **5**, 308–329.

R. B. Lehoucq (1995), Analysis and Implementation of an Implicitly Restarted Iteration, PhD thesis, Rice University, Houston, TX. Also available as Technical report TR95-13, Department of Computational and Applied Mathematics.

R. B. Lehoucq (2001), 'Implicitly restarted Arnoldi methods and subspace iteration', *SIAM J. Matrix Anal. Appl.* **23**, 551–562.

R. B. Lehoucq and A. G. Salinger (2001), 'Large-scale eigenvalue calculations for stability analysis of steady flows on massively parallel computers', *Internat. J. Numer. Methods Fluids* **36**, 309–327.

R. B. Lehoucq and J. A. Scott (1996), An evaluation of software for computing eigenvalues of sparse nonsymmetric matrices, Preprint MCS-P547-1195, Argonne National Laboratory, Argonne, IL.

R. B. Lehoucq and D. C. Sorensen (1996), 'Deflation techniques for an implicitly restarted Arnoldi iteration', *SIAM J. Matrix Anal. Appl.* **17**, 789–821.

R. B. Lehoucq, D. C. Sorensen and C. Yang (1998), *ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi methods*, SIAM Publications, Philadelphia, PA.

T. A. Manteuffel (1978), 'Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration', *Numer. Math.* **31**, 183–208.

K. J. Maschhoff and D. C. Sorensen (1996), P_ARPACK: An efficient portable large scale eigenvalue package for distributed memory parallel architectures, in *Applied Parallel Computing in Industrial Problems and Optimization*, Springer, Berlin, pp. 478–486.

K. Meerbergen (1996), Robust methods for the calculation of rightmost eigenvalues of nonsymmetric eigenvalue problems, PhD thesis, Katholieke Universiteit Leuven, Belgium.

K. Meerbergen and A. Spence (1997), 'Implicitly restarted Arnoldi with purification for the shift-invert transformation', *Math. Comput.* **218**, 667–689.

R. B. Morgan (1991), 'Computing interior eigenvalues of large matrices', *Lin. Alg. Appl.* **154–156**, 289–309.

R. B. Morgan (1996), 'On restarting the Arnoldi method for large nonsymmetric eigenvalue problems', *Math. Comput.* **65**, 1213–1230.

R. B. Morgan and D. S. Scott (1993), 'Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems', *SIAM J. Sci. Comput.* **14**, 585–593.

MPI Forum (1994), 'MPI: A Message-Passing Interface standard', *Internat. J. Supercomput. Appl. High Performance Comput.* Special issue on MPI. Electronic form: `ftp://www.netlib.org/mpi/mpi-report.ps`.

J. Olsen, P. Jørgensen and J. Simons (1990), 'Passing the one-billion limit in full configuration-interaction (FCI) calculations', *Chem. Phys. Lett.* **169**, 463–472.

C. C. Paige (1971), The computation of eigenvalues and eigenvectors of very large sparse matrices, PhD thesis, University of London.

C. C. Paige, B. N. Parlett and H. A. van der Vorst (1995), 'Approximate solutions and eigenvalue bounds from Krylov subspaces', *Numer. Lin. Alg. Appl.* **2**, 115–134.

B. N. Parlett (1980), *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ.

B. N. Parlett and J. K. Reid (1981), 'Tracking the progress of the Lanczos algorithm for large symmetric eigenproblems', *IMA J. Numer. Anal.* **1**, 135–155.

B. N. Parlett and D. Scott (1979), 'The Lanczos algorithm with selective orthogonalization', *Math. Comput.* **33**, 217–238.

L. Reichel (1990), 'Newton interpolation at Leja points', *BIT* **30**, 332–346.

A. Ruhe (1994*a*), 'The rational Krylov algorithm for nonsymmetric eigenvalue problems, III: Complex shifts for real matrices', *BIT* **34**, 165–176.

A. Ruhe (1994*b*), 'Rational Krylov algorithms for nonsymmetric eigenvalue problems, II: Matrix pairs', *Lin. Alg. Appl.* **197–198**, 283–295.

Y. Saad (1980), 'Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices', *Lin. Alg. Appl.* **34**, 269–295.

Y. Saad (1984), 'Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems', *Math. Comput.* **42**, 567–588.

Y. Saad (1992), *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK.

Y. Saad (1994), 'ILUT: A dual threshold incomplete LU factorization', *Numer. Lin. Alg. Appl.* **1**, 387–402.

J. A. Scott (200x), 'An Arnoldi code for computing selected eigenvalues of sparse real unsymmetric matrices', *ACM Trans. Math. Software*.

H. Simon (1984), 'Analysis of the symmetric Lanczos algorithm with reorthogonalization methods', *Lin. Alg. Appl.* **61**, 101–131.

V. Simoncini (1996), 'Ritz and pseudo-Ritz values using matrix polynomials', *Lin. Alg. Appl.* **241–243**, 787–801.

G. L. G. Sleijpen and H. A. van der Vorst (1995), 'An overview of approaches for the stable computation of hybrid BiCG methods', *Appl. Numer. Math.* **19**, 235–254.

G. L. G. Sleijpen and H. A. van der Vorst (1996), 'A Jacobi–Davidson iteration method for linear eigenvalue problems', *SIAM J. Matrix Anal. Appl.* **17**, 401–425.

D. C. Sorensen (1992), 'Implicit application of polynomial filters in a k-step Arnoldi method', *SIAM J. Matrix Anal. Appl.* **13**, 357–385.

D. C. Sorensen and C. Yang (1998), 'A truncated RQ-iteration for large scale eigenvalue calculations', *SIAM J. Matrix Anal. Appl.* **19**, 1045–1073.

A. Stathopoulos, Y. Saad and K. Wu (1998), 'Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods', *SIAM J. Sci. Comput.* **19**, 227–245.

G. W. Stewart (2001), 'A Krylov–Schur algorithm for large eigenproblems', *SIAM J. Matrix Anal. Appl.* **23**, 601–614.

W. J. Stewart and A. Jennings (1981), 'Algorithm 570: LOPSI, A Fortran subroutine for approximations to right or left eigenvectors corresponding to the dominant set of eigenvalues of a real symmetric matrix', *ACM Trans. Math. Software* **7**, 230–232.

L. N. Trefethen (1992), Pseudospectra of matrices, in *Numerical Analysis 1991* (D. F. Griffiths and G. A. Watson, eds), Longman, pp. 234–266.

L. N. Trefethen (1999), Computation of pseudospectra, in *Acta Numerica*, Vol. 9, Cambridge University Press, pp. 247–296.