UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Scalable Library for Eigenvalue Problem Computations SLEPc

SLEPc Technical Report **STR-2**
Available at http://slepc.upv.es

Single Vector Iteration Methods in SLEPc

V. Hernández
J. E. Román
A. Tomás
V. Vidal

Last update: June, 2005 (SLEPc 2.3.0)
Previous updates: –

Contents

1 Introduction

This report describes the algorithms for the solution of eigenvalue problems that are implemented in the SLEPc eigensolver `power` (`EPSPOWER`). This eigensolver covers several well-known single-vector iteration methods such as the Power Iteration, the Inverse Iteration and the Rayleigh Quotient Iteration. These algorithms were the first ones to be employed in the sparse eigenvalue scenario and come from the observation that a vector that is repeatedly multiplied by matrix A usually tends to align in the direction of the eigenvector associated to its dominant eigenvalue.

Normally, this eigensolver is not competitive with other methods provided in SLEPc, except maybe in some cases with the Rayleigh Quotient Iteration. Therefore, its use is not recommended. However, its simplicity makes it appropriate for helping understand how eigensolvers are implemented in SLEPc.

2 Description of the Methods

A brief description of the three methods and their variants follows below. Detailed theoretical analyses of convergence as well as several implementation aspects can be found in classical references or in more recent monographs, see for example [Wilkinson, 1965], [Parlett, 1980] or [Saad, 1992].

2.1 Direct and Inverse Iteration

The Direct Iteration (or Power Method) is the simplest single vector iteration method. It consists in premultiplying a vector by matrix A repeatedly, that is, generating the sequence of vectors $A^k v_0$ where v_0 is some nonzero initial vector.

Algorithm 1 (Basic Power Method)

```
Input: Matrix $A$ and initial vector $v_0$
Output: Approximate dominant eigenpair $(\theta, v)$
    $y = v_0$
    For $k = 1, 2, \ldots$
        $\theta = \|y\|_\infty$
        $v = y/\theta$
        $y = Av$
    end
```

In Algorithm 1 the vector is normalized at each iteration to avoid underflow and overflow. The advantage of the particular normalization used is that the approximation of the eigenvalue, θ, is readily available, which can then be used for the convergence test. If another norm is used for normalization then the eigenvalue must be computed explicitly as the Rayleigh quotient associated to the current approximation of the eigenvector, v. The Rayleigh quotient is defined as

$$R(v) = \frac{v^*Av}{v^*v} \quad . \tag{1}$$

The theory says that if A has a unique eigenvalue of largest modulus, λ_1, and v_0 is not deficient in the direction of the corresponding eigenvector, x_1, then the Power Iteration converges to a multiple of x_1. The rate of convergence is linear and depends on the factor $|\lambda_1/\lambda_2|$, being λ_2 the next largest eigenvalue in magnitude. Thus, convergence can be very slow if these two eigenvalue are close (in magnitude) and will not be achieved at all in the case that there is no unique dominant eigenvalue (e.g., λ_1 is part of a complex conjugate pair).

The Inverse Iteration consists simply in iterating with the matrix A^{-1} instead of the original matrix A. The method is often combined with shifts of origin so vectors are multiplied by $(A - \sigma I)^{-1}$, and the computed results are then the eigenvalue of A that is closest to scalar σ and the corresponding eigenvector. The main advantage of this method is a faster convergence due to a better separation of the eigenvalues, although it is still linear. The disadvantage is having to deal with the inverse. Fortunately, it is not necessary to compute the inverse explicitly. Instead, a linear system of equations is solved at each iteration, as shown in Algorithm 2.

Algorithm 2 (Inverse Iteration)

```
Input: Matrix $A$, value of the shift $\sigma$ and initial vector $v_0$
Output: Approximate eigenpair $(\theta, v)$
    $y = v_0$
    For $k = 1, 2, \ldots$
        $\theta = \|y\|_\infty$
        $v = y/\theta$
        Solve $(A - \sigma I)\, y = v$
    end
```

The linear system is usually solved with a direct method, where the factorization needs to be computed only once and is amortized in successive solves. The coefficient matrix $A - \sigma I$ is indefinite if σ is inside A's spectrum. In addition, it is nearly singular if σ is very close to an eigenvalue and, in this case, a large error should be expected in the solution computed by a direct solver. However, it can be shown that the error lies almost entirely in the direction of the wanted eigenvector and therefore it is harmless, [Parlett, 1980, §4.3].

An alternative to the direct linear solver is to use an iterative linear solver. Eigensolvers that make use of iterative linear solvers are generally referred to as *inexact* eigensolvers or *inner-outer* iterations. Typically, it is necessary to use a more stringent convergence tolerance for the linear solver than for the eigensolver itself. Also, this scheme is not recommended if σ is very close to an eigenvalue because that implies a very slow convergence of the inner iteration.

2.2 Rayleigh Quotient Iteration

Analysis of Inverse Iteration shows that convergence is faster if σ is much closer to λ_1 than it is to λ_2. Therefore, an improved method is to change the shift σ occasionally into a value that is known to be a better approximation of the eigenvalue λ_1 than the previous σ. One possibility is to take the new shift to be the Rayleigh quotient of the last approximate eigenvector, resulting in the Rayleigh Quotient Iteration (RQI), shown in Algorithm 3.

Algorithm 3 (Rayleigh Quotient Iteration, RQI)

```
Input: Matrix $A$ and initial vector $v_0$
Output: Approximate eigenpair $(\theta, v)$
    $y = v_0$
    $v = y/\|y\|_2$
    Choose $\sigma_1$ or compute it as $\sigma_1 = R(v)$
    For $k = 1, 2, \dots$
        Solve $(A - \sigma_k I)\, y = v$
        $\gamma = y^* v$
        $\delta = \|y\|_2$
        $\sigma_{k+1} = \sigma_k + \gamma/\delta^2$
        $v = y/\delta$
    end
    $\theta = \sigma_{k+1}$
```

In the above algorithm, σ_{k+1} is nothing more than the Rayleigh quotient, computed efficiently:

$$\sigma_{k+1} = \sigma_k + \frac{\gamma}{\delta^2} = \sigma_k + \frac{y^* v}{y^* y} = \sigma_k + \frac{y^*(A - \sigma_k I)\, y}{y^* y} = \frac{y^* A\, y}{y^* y} \quad . \tag{2}$$

In other words, the new shift is computed by updating the previous one with a correction that corresponds to the Rayleigh quotient of the shifted matrix. It can be shown that δ is the reciprocal of the residual norm $\|r\|_2 = \|Av - \sigma_k v\|_2$ for the approximated eigenpair (σ_k, v). So the convergence criterion can also be implemented very efficiently.

If matrix A is Hermitian then RQI has locally cubical convergence, [Wilkinson, 1965, ch. 9, §62]. That is, once the current estimate (σ_k, v) becomes sufficiently close to some eigenpair, then each iteration triples the number of accurate digits. If A is non-Hermitian then it is still possible to get cubical convergence if using a two-sided version of the algorithm (see below).

One drawback of RQI is that it may converge to an eigenvalue which is not the closest to the initial shift σ_1. This could be remedied by initially performing a few iterations of inverse iteration, [Szyld, 1988]. Another drawback of RQI is its higher cost since it requires a factorization at every iteration. Again, solving the linear systems iteratively may be an acceptable alternative.

2.3 Variations of the Algorithms

Practical implementations often introduce some of the variations described next in order to improve the usability of the algorithms.

Wilkinson shift. In Algorithm 3, it is sometimes possible to improve convergence by choosing a new shift different from the Rayleigh quotient: the Wilkinson shift. This shift was proposed in the context of LR and QR iterations, but the same idea can be applied to RQI, see [Parlett, 1980, §8.10]. The computation of this shift requires an additional matrix-vector product, so usually the potential reduction in the number of iterations does not pay off the extra effort.

Two-sided versions. It is possible to formulate two-sided versions of the algorithms. In general, two-sided eigensolvers are those which try to compute approximations of both left and right invariant subspaces. In the context of the above algorithms, this means computing also the left eigenvector, w, such that $w^*A = w^*\theta$. This makes sense only in the case of non-Hermitian problems since otherwise left and right eigenvectors coincide.

The left eigenvector is an eigenvector of A^* with eigenvalue $\bar{\theta}$. This fact can be exploited to define two-sided versions of direct and inverse iteration, where two sequences of vectors are generated: one multiplying by A to get v and another multiplying by A^* to get w. In the case of RQI, in addition, it is possible to make use of the generalized Rayleigh quotient of A at v and w,

$$R(v,w) = \frac{w^*Av}{w^*v} \quad . \tag{3}$$

This quantity represents a more accurate approximation of the eigenvalue than the ordinary Rayleigh quotient (1), assuming that v and w are the approximate right and left eigenvectors, respectively. Due to this good property, two-sided RQI (Algorithm 4) retains the locally cubical convergence condition in the non-Hermitian case. Further details about the algorithm can be found in [Ostrowski, 1959] and [Parlett, 1974].

Algorithm 4 (Two-sided RQI)

Input: Matrix A and initial vectors v_0, w_0
Output: Approximate eigentriple (θ, v, w)

```
$y = v_0$, $v = y/\|y\|_2$
$z = w_0$, $w = z/\|z\|_2$
Choose $\sigma_1$ or compute it as $\sigma_1 = R(v,w)$
For $k = 1, 2, \ldots$
    Solve $(A - \sigma_k I)\, y = v$
    Solve $(A - \sigma_k I)^*\, z = w$
    $\gamma = z^* v$
    $\delta = \sqrt{z^* y}$
    $\sigma_{k+1} = \sigma_k + \gamma/\delta^2$
    $v = y/\delta$, $w = z/\delta$
end
$\theta = \sigma_{k+1}$
```

Deflation. Assuming that an eigenvalue λ_1 and its corresponding eigenvector x_1 have been computed with any of the single-vector iteration methods described so far, the same algorithm can be rerun to compute the next ones, provided that it is equipped with some form of *deflation* to avoid convergence to the previously computed eigenpair.

A general deflation procedure is to apply a rank one modification to matrix A so as to displace eigenvalue λ_1 while keeping all other eigenvalues unchanged,

$$A_1 = A - \sigma x_1 v^* \quad . \tag{4}$$

Assuming that the eigenvector x_1 has unit norm and v is any vector satisfying $v^* x_1 = 1$, matrix (4) has the same eigenvalues as A except for the eigenvalue λ_1 which is transformed into $\lambda_1 - \sigma$. The natural value for σ is λ_1. For vector v, there are two possible choices that are usually considered. The first one is to use $v = x_1$, the computed eigenvector. This choice has the advantage of preserving the eigenvectors in addition to eigenvalues. The other choice is $v = y_1$, the left eigenvector computed by a two-sided eigensolver. In this case, the transformation preserves both right and left eigenvectors.

Deflation based on equation (4) can be easily implemented without building A_1 explicitly. However, as suggested by Wilkinson, it is not recommended due to potential stability problems. A preferred deflation procedure, which is mathematically equivalent, consists in purging the iterate vector of all previously converged eigenvectors. Suppose we want to perform a matrix-vector product with matrix A_1,

$$A_1 x = (A - \lambda_1 x_1 x_1^*)x = (A - A x_1 x_1^*)x = A(I - x_1 x_1^*)x = A\tilde{x} \quad , \tag{5}$$

where $\tilde{x} = (I - x_1 x_1^*)x$ is the result of orthogonalizing x with respect to x_1, or applying the orthogonal projector represented by matrix $P = I - x_1 x_1^*$. The idea extends easily to more than one eigenvector. In the context of two-sided methods, it is also possible to define this form of deflation, but with the analog concepts of bi-orthogonalization and oblique projection.

In [Dax, 2003], a selective orthogonalization criterion is proposed in the context of RQI, trying to avoid unnecessary orthogonalizations. With this criterion, the current vector is orthogonalized only with respect to approximate eigenvectors x_i whose associate eigenvalue satisfies $|\lambda_i - \sigma_k| \leq k\|A\|_\infty/1000$.

Definite pencils. In order to solve a generalized eigenvalue problem with the above algorithms, a spectral transformation has to be used. In the case of Hermitian positive-definite problems, symmetry is lost in the transformed operator, $B^{-1}A$, $(A - \sigma B)^{-1}B$, etc. Then, a two-sided algorithm is required to retain locally cubical convergence. A better alternative could be to reorient the algorithms so that they are based on the Rayleigh quotient of a matrix pair,

$$R_{A,B}(v) = \frac{v^*Av}{v^*Bv} \quad . \tag{6}$$

It is easy to see that the meaning of all the quantities of the algorithms is retained if we replace the standard Hermitian inner product, $\langle x, y\rangle = y^*x$, by the B-inner product, $\langle x, y\rangle_B = y^*B\,x$.

Block versions. The Power iteration can be generalized to operate on a set of vectors instead of a single vector. This new procedure is usually referred to as subspace iteration or simultaneous iteration. Details related to this procedure can be found in SLEPc Technical Report STR-3, “Subspace Iteration in SLEPc”. With respect to RQI, block versions have also been proposed, see for instance [Absil *et al.*, 2002].

2.4 Survey of Available Implementations

Due to their simplicity, single vector iteration methods have been traditionally implemented by the users themselves. For this reason, publicly available implementations are scarce.

An old implementation of the Power Iteration is available in NAPACK (`http://www.netlib.org/napack/power.f`). It handles complex conjugate eigenvalues by a simple block technique with two vectors. PEIGS (`http://www.emsl.pnl.gov/docs/nwchem`) provides an implementation of a variant of inverse iteration tailored for dense symmetric eigenproblems. Finally, direct, inverse and Rayleigh quotient iterations are implemented in IETL (`http://www.comp-phys.org/software/ietl`), as well as in the Java library SMT (`http://www.math.uib.no/~bjornoh/mtj/smt`).

3 The SLEPc Implementation

The SLEPc solver `EPSPOWER` provides an implementation of most of the algorithms and variants described in the previous section.

When using the default spectral transformation (`STSHIFT`), this solver provides an implementation of the Power Iteration, whereas in shift-and-invert mode (`STSINV`) the resulting method is the Inverse Iteration. In the latter case, by default the shift is constant. An option described below allows the user to specify a variable shift thus turning to an RQI variant.

The implemented algorithms incorporate deflation (when requesting more than one eigenvalue), two-sided versions (with `EPSSetClass(eps,EPS_TWO_SIDE)`), and support for definite pencils (when the problem type has been set to `EPS_GHEP`).

3.1 User Options

The user is able to change the way in which the shift σ is updated:

```
EPSPowerSetShiftType(EPS eps,EPSPowerShiftType type)
```

This function is only relevant in shift-and-invert mode (`STSINV`). By default, the shifts are constant. The other possibilities are `EPSPOWER_SHIFT_RAYLEIGH` and `EPSPOWER_SHIFT_WILKINSON`. This can also be changed with the command-line option `-eps_power_shift_type`. Another related function is:

```
STSetShift(ST st,PetscScalar shift)
```

This function is not specific to the `power` solver, and it changes the value of the shift, σ, in the associated `ST` object. It is used in the implementation of the RQI method to set the initial shift. This can also be set with the command-line option `-st_shift`. Note that calling this function may involve refactorization of the matrix $A - \sigma B$.

3.2 Known Issues and Applicability

Single-vector iteration methods provided in SLEPc cannot cope with complex conjugate eigenvalue pairs when working in real arithmetic. As stated above, the Power Iteration does not converge in this case. However, a simple analysis shows that the subspace spanned by two consecutive iterates contains converging approximations to the complex pair of eigenvectors. A simple block version working with two vectors would be able to extract the desired eigenvalues and eigenvectors. However, this possibility is not taken into account in SLEPc in order to keep the implementation simple. In order to solve problems of this type, other solvers such as `subspace` should be used.

This solver can only be used for the case that the wanted eigenvalues are those of largest magnitude (see `EPSWhichEigenpairs`).

The Power Iteration can be applied to any PETSc matrix object, including shell matrices provided that the `MatMult` operation is implemented (as well as `MatMultTranspose` in the two-sided solver). In the case of Inverse Iteration and RQI, then the behavior is not so simple because a linear system of equations must be solved, thus possibly requiring other operations such as `MatAXPY` or `MatGetDiagonal`. There is also the possibility that the linear solve fails if the shift σ happens to be too close to an eigenvalue.

In the two-sided algorithm, it is not guaranteed that the solver is able to compute more than one eigenpair.

Supported problem types	All
Allowed portion of the spectrum	Largest $\lvert\lambda\rvert$
Support for complex numbers	Yes
Support for left eigenvectors	Yes

References

Absil, P.-A., P. V. Dooren, R. Mahony, and R. Sepulchre (2002). A Grassmann-Rayleigh Quotient Iteration for Computing Invariant Subspaces. *SIAM Rev.*, 44(1):57–73.

Dax, A. (2003). The Orthogonal Rayleigh Quotient Iteration (ORQI) Method. *Linear Algebra Appl.*, 358(1–3):23–43.

Ostrowski, A. M. (1959). On the Convergence of the Rayleigh Quotient Iteration for the Computation of the Characteristic Roots and Vectors. III (Generalized Rayleigh Quotient and Characteristic Roots with Linear Elementary Divisors). *Arch. Rational Mech. Anal.*, 3:325–240.

Parlett, B. N. (1974). The Rayleigh Quotient Iteration and Some Generalizations for Nonnormal Matrices. *Math. Comp.*, 28:679–693.

Parlett, B. N. (1980). *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, NJ. Reissued with revisions by SIAM, Philadelphia, 1998.

Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems: Theory and Algorithms*. John Wiley and Sons, New York.

Szyld, D. (1988). Criteria for Combining Inverse Iteration and Rayleigh Quotient Iteration. *SIAM J. Numer. Anal.*, 25(6):1369–1375.

Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*. Claredon Press, Oxford, UK.