Numerical Solution of Large Nonsymmetric Eigenvalue Problems [*]

Youcef Saad
RIACS
MS 230-5, NASA Ames Research Center
Moffett Field, CA 94035

Abstract

We describe several methods based on combinations of Krylov subspace techniques, deflation procedures and preconditionings, for computing a small number of eigenvalues and eigenvectors or Schur vectors of large sparse matrices. The most effective techniques for solving realistic problems from applications are those methods based on some form of preconditioning and one of several Krylov subspace techniques, such as Arnoldi's method or the Lanczos procedure. We consider two forms of preconditionings: shift-and-invert and polynomial acceleration. The latter presents some advantages for parallel / vector processing but may be ineffective if eigenvalues inside the spectrum are sought. We provide some algorithmic details that improve the reliability and effectiveness of these techniques.

Key words: Sparse eigenvalue problems; Krylov subspace methods; Arnoldi's method; Lanczos Algorithm; Polynomial preconditioning; Deflation procedures; Shift and invert.
Abbreviated title: Solution of large eigenvalue problems.

[*]This work was supported in part by USRA under NASA Grant Number NCC 2-387 and in part by ARO under contract DAAL03-88-K-0085.

1 Introduction

Many important problems in science and engineering require the computation of a small number of eigenvalues with algebraically largest (or smallest) real parts of a large nonsymmetric real matrix A. Among the typical examples from the literature, see e.g., [9], we only mention the important class of stability analysis and more generally of bifurcation problems [17], from which we will draw our main test example. From the numerical point of view, nonsymmetric eigenvalue problems can be substancially more difficult to solve than the symmetric ones. This is due to the fact that eigenvalues of large matrices can be arbitrarily poorly conditioned. In this paper we will propose a few techniques and tools that can be combined with the traditional projection methods to enhance their efficiency and robustness.

There have been mainly three basic projection methods for solving large nonsymmetric eigenvalue problems investigated so far. The first is Bauer's subspace iteration method and its many variations [2, 7, 15, 16, 41, 42, 45]. An important drawback of this method, recognized both in the symmetric case [23, 22], and the nonsymmetric case [29, 32] is that it may be exceedingly slow to converge. Another known weakness is that it computes the dominant eigenvalues of A, i.e., those having largest modulii, whereas in many important applications it is the eigenvalues with largest real parts that are wanted. However, this difficulty can be obviated by using Chebyshev acceleration as is suggested in [32]. The second method is due to Arnoldi [1, 29] and is essentially an orthogonal projection method on the Krylov subspace $\{v_1, Av_1, \ldots A^{m-1}v_1\}$. Thus, Arnoldi's method is a generalization of the symmetric Lanczos algorithm. Its main drawback is that, unlike the symmetric Lanczos algorithm, the growth of computational time and storage becomes excessively high as the number of steps increases. Variations on the basic scheme have been proposed [29], which lead to oblique projection type techniques [30], but their theory is not well understood. Finally, the third method is the nonsymmetric Lanczos method [8, 20, 25, 26, 43] which is another generalization of the symmetric Lanczos algorithm due originally to Lanczos. It produces a tridiagonal matrix some eigenvalues of which can be taken as approximations to the eigenvalues of A. At the difference with Arnoldi's method, this is not an orthogonal projection method, but an oblique projection method [30]. The algorithm runs the risk of breaking down at any step, which has given a poor reputation to the method in the past [45]. Parlett, Taylor and Liu [26] have suggested an elegant solution to this problem. Cullum and Willoughby [8] have extended their symmetric Lanczos algorithm without reorthogonalization, to the nonsymmetric case and suggest a new way for dealing with the resulting non-hermitian tridiagonal matrices. On the whole the main difficulty with the nonsymmetric Lanczos method is theoretical, as the method is not too well understood.

To these three basic methods one should add a number of tools such as deflation processes and acceleration/ preconditioning techniques the best example of which is the so-called shift-and-invert strategy. This paper is not concerned with the projection methods enumerated above but rather with the implementation of these secondary tools. It shows their importance and proposes ways to put together efficient methods that exploits them in conjunction with the projection techniques. More specifically we describe:

- A particular case of Wielandt deflation which is of interest when computing Schur vectors. We refer to this as Schur-Wielandt deflation.

- A shift-and-invert strategy for general nonhermitian matrices.

- A polynomial preconditioning technique consisting of iterating with the matrix $p(A)$, where p is a carefully chosen low degree polynomial.

These techniques can be combined with any of the three projection methods discussed above but we will illustrate their implementation only with Arnoldi's method for the sake of brievety.

Before proceeding further we would like to point out a few key differences between the three projection methods. First, we emphasize that, in practice, subspace iteration is only able to compute a small number of eigenvalues and associated eigenvectors of a large nonsymmetric matrix. To some extent Arnoldi's method presents the same limitations in practice. The nonsymmetric Lanczos algorithm without reorthogonalization or with some form of partial reorthogonalization is the only method that has the potential of computing a large number of eigenvalues and eigenvectors of a nonsymmetric matrix A [8, 25]. On the other hand the Lanczos algorithm requires the use of both the matrix A and of its transpose. In some applications the matrix A is not available explicitly but the action of multiplying A by a vector is easy to perform, by use of a finite difference formula. In those cases A^T is not available and cannot even be approximated with finite differencing. For example one may be interested in studying the stability of a dynamical system governed by a partial differential equation of the form

$$\frac{\partial u}{\partial t} = F(u, \theta) \tag{1}$$

where F is a partial differential operator, and θ some real parameter, as θ varies. Such a system is said to be stable if all the eigenvalues of the Jacobian of F with respect to u, computed at the steady state solution, have negative real parts. All that may be wanted here is to compute one eigenvalue or a complex pair of eigenvalues. Although the Jacobian matrix J at some coordinate u may not be available explicitly, the multiplication of J by an arbitrary vector x can be carried out, usually at low cost, with the help of the difference formula

$$Jx \approx \frac{F(u + \epsilon x, \theta) - F(u, \theta)}{\epsilon}, \tag{2}$$

where ϵ is some small and carefully chosen scalar.

The approximation (2) has been useful in solving nonlinear systems of equations [3, 6, 12, 4, 44, 19] and to compute eigenvalues of various semi-discrete operators [11] used in compressible fluid flow calculations. Here, an algorithm such as Arnoldi's method can be used but not the nonsymmetric Lanczos procedure since we do not know to compute the vector $J^T x$ for any vector x when the Jacobian matrix J is not explicitly available.

In the numerical experiments section we illustrate these techniques with an example issued from a well-known and simple bifurcation model from Chemical engineering. Problems of this type are numerous in structural engineering [5], in aerodynamics (the panel flutter problem [37]), chemical engineering [14], fluid mechanics [18] and many other fields.

2 A Schur-Wielandt deflation technique

When used with caution, deflation procedures can be quite useful and effective if a small number of eigenvalues and eigenvectors are to be computed. In the nonsymmetric case many common deflation techniques require the knowledge of both right and left eigenvectors. These procedures, an example of which is Hotelling's deflation, can be ill-conditioned if only because the determination of eigenvectors of a general sparse matrix can be itself untrustworthy. In fact in the defective case there may not exist a basis of the invariant subspace consisting of eigenvectors and therefore any numerical method that attempts to determine such a basis will have numerical difficulties. As suggested by Stewart [42] it is preferable to work with Schur vectors, i.e. with an orthonormal basis of the invariant subspace, when dealing with the nonsymmetric eigenvalue problem. A partial Schur factorization is of the form

$$AU = UR \tag{3}$$

where U is an N x p complex orthogonal matrix ($U^H U = I$) and R is upper triangular complex matrix. Here, X^H denotes the transpose of the complex conjugate of a matrix X. Note that the order of the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_p$ as they appear in the upper triangular matrix R is crucial. In fact, when the eigenvalues $\lambda_1, \lambda_2, ..\lambda_p$ are distinct, then for a given order this factorization is unique in the usual sense of QR factorizations, i.e. the columns of U are uniquely determined up to a sign of the form $e^{i\theta}$. Thus, whenever we choose a certain ordering of the eigenvalues, we can deal with the Schur vectors without confusion in the same way that we deal with the eigenvectors of a Hermitian matrix. We will consider later the problem of avoiding complex arithmetic when the matrix A is real.

In this section we describe a deflation technique which is a simple variation of Wielandt's deflation and show how it can be put to work to compute an orthonormal basis of an invariant subspace and the corresponding partial Schur forms. We start our discussion with the general Wielandt deflation with one vector and explain why Schur-Wielandt deflation is often nearly optimal. In the following we denote by $||.||_2$ the 2-norm in $\mathbf{C}^N$. Unless otherwise stated the eigenvalues are ordered in decreasing order of their real parts (if a conjugate pair occur then the one with positive imaginary part is first). All eigenvectors are assumed to be normalized by their Euclidean norms.

2.1 Wielandt deflation with one vector

Suppose that we have computed the eigenvalue λ_1 of algebraically largest real part and a corresponding eigenvector u_1 of a matrix A by some basic algorithm, say algorithm (A), which delivers the eigenvalue of largest real part of the input matrix, along with an eigenvector. For example, if the matrix A is known to have real eigenvalues, algorithm (A) can be some variant of the power method. It is assumed in what follows that the vector u_1 is normalized so that $||u_1||_2 = 1$. The problem is to compute the next eigenvalue λ_2 of A. An old technique for achieving this is what is commonly called a deflation procedure: a rank one modification of the original matrix is performed so as to displace the eigenvalue λ_1, while keeping all other eigenvalues unchanged. The rank one modification is chosen so that the eigenvalue λ_2 becomes the one with largest real part of the modified matrix and therefore, Algorithm (A) can again be applied to the new matrix to retrieve the pair λ_2, u_2.

In the general procedure known as Wielandt's deflation only the knowledge of the right eigenvector is required. The deflated matrix is of the form

$$A_1 = A - \sigma u_1 v^H \tag{4}$$

where v is an arbitrary vector such that $v^H u_1 = 1$, and σ is an appropriate shift. As is well-known the eigenvalues of A_1 are the same as those of A except for the eigenvalue λ_1 which is transformed into the eigenvalue $\lambda_1 - \sigma$.

Theorem 2.1 *(Wielandt): The spectrum of A_1 is*

$$\sigma(A_1) \;=\; \{\lambda_1 - \sigma,\; \lambda_2, \lambda_3, ..., \lambda_p\} \tag{5}$$

Moreover, the left eigenvectors of A associated with $\lambda_2, \lambda_3, \ldots, \lambda_N$ are preserved under the deflation process, and so is the right eigenvector u_1.

It is important to determine what the right eigenvectors become when $i \neq 1$. For each i, we will seek a right eigenvector of A_1 in the form of $\hat{u}_i = u_i - \gamma_i u_1$. We have

$$A_1 \hat{u}_i = (A - \sigma u_1 v^H)(u_i - \gamma_i u_1) = \lambda_i u_i - [\gamma_i \lambda_1 + \sigma(u_i, v) - \sigma \gamma_i] u_1 \tag{6}$$

When $i = 1$, taking $\gamma_1 = 0$ shows again that any eigenvector associated with the eigenvalue λ_1 remains an eigenvector of A_1, associated with the eigenvalue $\lambda_1 - \sigma$. For $i \neq 1$, it is possible to choose γ_i so that the vector $\hat{u}_i$ is an eigenvector of A_1 associated with the eigenvalue λ_i:

$$\gamma_i = \frac{\sigma\ (u_i, v)}{\sigma - (\lambda_1 - \lambda_i)} \tag{7}$$

Observe that the above expression is not defined when the denominator vanishes, but it is then known that the eigenvalue $\lambda_i = \lambda_1 - \sigma$ is already an eigenvalue of A_1, i.e., the eigenvalue $\lambda_1 - \sigma$ becomes multiple, and we only have one eigenvector namely u_1.

There are infinitely many different ways of choosing the vector v which can be taken to be any vector in the affine subspace of dimension $N - 1$ of vectors whose inner product with the vector u_1 is equal to one. A common choice is to take $v = w_1$ the left eigenvector. This is referred to as Hotelling's deflation and has the advantage of preserving both the left and right eigenvectors of A as is seen from the fact $\gamma_i = 0$ in this situation.

An interesting question that remains to be answered is the following: *among all the choices of v, which one(s) will provide the best possible condition number for the next eigenvalue λ_2 to be computed?* The condition number of an eigenvalue is defined as the inverse of the cosine of the angle between its corresponding right and left eigenvectors. It is a measure of the sensitivity of the eigenvalue to perturbations in A. Thus, a large condition number, i.e., a poorly conditioned eigenvalue, will cause difficulties to the numerical algorithm that is used to compute that eigenvalue. This consitutes the motivation for the above question.

We will distinguish the eigenvalues and eigenvectors associated with the matrix A_1 from those of A by denoting them with a tilde. The condition number of the next eigenvalue $\tilde{\lambda}_2$ to be computed is, by definition [45, 13],

$$Cond(\tilde{\lambda}_2) = \frac{||\tilde{u}_2||_2 ||\tilde{w}_2||_2}{|(\tilde{u}_2, \tilde{w}_2)|} \tag{8}$$

where $\tilde{u}_2, \tilde{w}_2$ are the right and left eigenvectors of A_1 associated with the eigenvalue λ_2. From what we have seen earlier, we know that $\tilde{w}_2 = w_2$ while $\tilde{u}_2 = u_2 - \gamma_2 u_1$ where γ_2 is given by (7). Assuming that $||w_2||_2 = 1$ we get,

$$Cond(\tilde{\lambda}_2) = \frac{||u_2 - \gamma_2 u_1||_2}{|(u_2, w_2)|} \tag{9}$$

where we have used the fact that $(u_1, w_2) = 0$. It is then clear from (9) that the condition number of λ_2 is minimized when $\gamma_2 = (u_2, u_1) \equiv cos\theta(u_2, u_1)$. Let us define $z = u_2 - cos\theta(u_2, u_1)u_1$, the vector obtained by orthogonalizing u_2 against u_1. Substituting this result in (7) we obtain

$$(z + cos(\theta(u_2, u_1))u_1, v) = \frac{\sigma \ (u_2, v)}{\sigma - (\lambda_1 - \lambda_2)} \tag{10}$$

which yields the condition,

$$(z, v) = -\frac{1}{\sigma}(\lambda_1 - \lambda_2) \ cos\theta(u_1, u_2) \tag{11}$$

to which we must add the normalizing constraint,

$$(u_1, v) = 1 \tag{12}$$

There are still infinitely many vectors v that satisfy the above two conditions in general. Condition (11) may seem impractical because it uses the knowledge of the right eigenpair (λ_2, u_2) which we are precisely trying to compute but one can think of an adaptive procedure in which the deflation is adjusted as the iteration process delivers better approximations of (λ_2, u_2). Moreover, we are interested from the theoretical point of view, in analyzing how far from optimal are the common choices of v. The conditions (11) and (12) allow us to select v in a linear space of dimension 2. We will consider two important choices:

(1) Choose v in the linear span of u_1 and u_2;

(2) Choose v in the linear span of u_1 and w_1.

Consider (1) first. In this case we find that the optimal v is

$$v_{opt} \ = \ u_1 - \frac{1}{\sigma}(\lambda_1 - \lambda_2) \ cotan\theta(u_2, u_1)\hat{z} \tag{13}$$

in which $\hat{z} = z/||z||_2$. We also find that

$$Cond(\tilde{\lambda}_2) = Cond(\lambda_2) \ sin\theta(u_2, u_1) \tag{14}$$

Not surprisingly, we note that when θ is close to $\pi/2$, the choice $v = u_1$ is nearly optimal. This is the situation of the hermitian case. Moreover, when $(\lambda_2 - \lambda_1)$ is small with respect to σ and the angle $\theta(u_2, u_1)$ is not too small, then the choice $v = u_1$ is again nearly optimal. This particular choice has an interesting additional property: it preserves the Schur vectors.

Proposition 2.1 *Let u_1 be an eigenvector of A of norm 1, associated with the eigenvalue λ_1 and let $A_1 \equiv A - \sigma u_1 u_1^H$. Then the eigenvalues of A_1 are $\tilde{\lambda}_1 = \lambda_1 - \sigma$ and $\tilde{\lambda}_j = \lambda_j, j = 2, 3..., N$. Moreover, the Schur vectors associated with $\tilde{\lambda}_j, j = 1, 2, 3..., N$ are identical with those of A.*

Proof. Let $AU = UR$ be the Schur factorization of A, where R is upper triangular and U is orthonormal. Then we have

$$A_1 U = [A - \sigma u_1 u_1^H]U = UR - \sigma u_1 e_1^H = U[R - \sigma e_1 e_1^H] \tag{15}$$

The result follows immediately. □

We now turn to the second way of choosing v. In this case we can express v as $v = \alpha u_1 + \beta w_1$ and the optimality condition substituted in (7) yields,

$$\frac{(u_2, \alpha u_1 + \beta w_1)}{1 - (\lambda_1 - \lambda_2)/\sigma} = (u_2, u_1) \tag{16}$$

Because w_1 is orthogonal to u_2 we immediately get

$$\bar{\alpha} = 1 - \frac{\lambda_1 - \lambda_2}{\sigma} \tag{17}$$

and from the constraint (12),

$$\beta = \frac{1}{(w_1, u_1)}(1 - \alpha) \tag{18}$$

yielding the optimal vector

$$v_{opt} = (1 - \frac{\bar{\delta}_2}{\bar{\sigma}})u_1 + \frac{\bar{\delta}_2}{\bar{\sigma}(w_1, u_1)} w_1 \tag{19}$$

where we have set for convenience $\delta_2 = \lambda_1 - \lambda_2$. Moreover, the new condition number $Cond(\tilde{\lambda}_2)$ is also given by (14). It is revealed by (19) that when the first eigenvector is well conditioned then since δ_2 is usually much smaller than σ, v_{opt} will not be too different from u_1. On the other hand if the first eigenvalue is very ill conditioned, the best v is closer to w_1. An interesting difference with the previous case is that everything here is computable except δ_2 which can be easily estimated dynamically. Apart from this practical difference, the two methods will provide the same condition number, i.e., they are, from the qualitative point of view, identical. However, the second method requires computing both the right and left eigenvectors whereas the first only requires the right eigenvector.

2.2 Deflation with a block of vectors

Let $u_1, u_2, \ldots u_j$ be a set of Schur vectors associated with the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_j$. We denote by U_j the matrix of column vectors $u_1, u_2, \ldots u_j$. Thus,

$$U_j \equiv [u_1, u_2, \ldots, u_j] \tag{20}$$

is an orthonormal matrix whose columns form a basis of the eigenspace associated with the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_j$. We do not assume here that these eigenvalues are real, so the matrix U_j may be complex. An immediate generalization of Proposition (2.1) is as follows.

Proposition 2.2 *Let Σ_j be the p x p diagonal matrix $\Sigma_j = Diag\{\sigma_1, \sigma_2, \ldots \sigma_j\}$. Then the eigenvalues of the matrix*

$$A_j \equiv A - U_j \Sigma_j U_j^H, \tag{21}$$

are $\lambda_i' = \lambda_i - \sigma_i$ for $i \leq j$ and $\lambda_i' = \lambda_i$ for $i > j$. Moreover, its associated Schur vectors are identical with those of A.

Proof. Let $AU = U$ be the Schur factorization of A. Thus, the matrix of the first j columns of U being equal to U_j. We have

$$A_j U = [A - U_j \Sigma_j U_j^H] U = UR - U_j \Sigma_j E_j^H, \tag{22}$$

where $E_j = [e_1, e_2, \dots e_j]$. Hence

$$A_j U = U[R - E_j \Sigma E_j^H] \tag{23}$$

and the result follows. □

2.3 Explicit Schur-Wielandt deflation in practice

It is interesting to note that the preservation of the Schur vectors is analogous to the preservation of the eigenvectors under Hotelling's deflation in the hermitian case. The above results suggest a very simple incremental deflation procedure consisting of building the matrix U_j one column at a time. Thus, at the j-th step, once the eigenvector $\tilde{u}_{j+1}$ of A_j is computed by the appropriate algorithm (A) we can orthonormalize it against all previous u_i's to get the next Schur vector u_{j+1} which will be appended to u_j to form the new deflation matrix U_{j+1}. It is a simple exercise to show that the vector u_{j+1} thus computed is a Schur vector associated with the eigenvalue λ_{j+1} and therefore at every stage of the process we have the desired decomposition

$$AU_j = U_j R_j, \tag{24}$$

where R_j is some $j \times j$ upper triangular matrix.

More precisely we may consider the following algorithm, in which the successive shifts σ_i are chosen so that for example $\sigma_i = \lambda_i$.

Algorithm

```
Do $i = 0, 1, 2, ..., j-1$:
1. Define $A_i \equiv A_{i-1} - \sigma_{i-1} u_{i-1} u_{i-1}^H$ (initially define $A_0 \equiv A$) and compute the dominant eigenvalue $\lambda_i$ of $A_i$ and the corresponding eigenvector $\tilde{u}_i$.
2. Orthonormalize $\tilde{u}_i$ against $u_1, u_2, ..., u_{i-1}$ to get the vector $u_i$.
```

From the practical point of view there are a few important details that make the deflation procedure more effective. We first address the question of avoiding complex arithmetic when the matrix A is real. With the above implementation, we may have to perform most of the computation in complex arithmetic even when A is real. Fortunately, when the matrix A is real, this can be avoided. In this case the Schur form is traditionally replaced by the quasi-Schur form, in which one still seeks the factorization (11) but simply requires that the matrix R_j, be quasi-triangular, i.e. one allows for 2×2 diagonal blocks. In practice, if λ_{j+1} is complex, most algorithms do not compute the complex eigenvector y_{j+1} directly but rather deliver its real and imaginary parts y_R, y_I separately. Thus the two eigenvectors $y_R \pm i y_I$ associated with the complex pair of conjugate eigenvalues $\lambda_{j+1}, \lambda_{j+2} = \bar{\lambda}_{j+1}$ are obtained at once.

Thinking in terms of bases of the invariant subspace instead of eigenvectors, we note that the real and imaginary parts of the eigenvector, generate the same subspace as the two conjugate eigenvectors and therefore there is no point in working with the (complex) eigenvectors instead of these two real vectors. Hence if a complex pair occurs, we only have to orthogonalize the two vectors y_R, y_I against all previous u_i's and pursue the algorithm in the same way. The only difference is that the size of U_j increases by two instead of just one in these instances.

Another practical consideration, is that one never needs to form A_1 explicitly. This is important because in general this matrix is full. If the algorithm we are using requires only matrix by vector multiplications, then clearly an operation of the form $y = A_1x = (A - \sigma uv^H)x$ can be performed as follows

a. Form $y := Ax$ and the scalar $t = \sigma\ (x, v)$

b. Compute $y := y - t\ u_1$

This procedure only requires that the vectors u_1, and v be kept in memory along with the matrix A. It is possible to deflate A_1 again into A_2 , and then into A_3 etc. At each step of the process we have

$$A_i = A_{i-1} - \sigma \tilde{u}_i v_i^H \tag{25}$$

Here one needs only to save the vectors $\tilde{u}_i$ and v_i along with the matrix A. However, one should be careful about the usage of deflation in general. It should not be used to compute more than a few eigenvalues and vectors. This is especially true in the non hermitian case because of the fact that the matrix A_i will accumulate errors from all previous computations and this could be disastrous if the eigenvalue comput ed is poorly conditioned. A careful analysis of these accumulation of errors is proposed in the Appendix, where a computable upper bound for the error on the computed invariant subspace is proposed. Another serious difficulty with deflating with a large number of vectors is the high computational cost.

We now give a sketch of the Schur-Wielandt deflation procedure for computing the p eigenvalues of largest real parts of a matrix A.

Algorithm: Explicit Schur-Wielandt Deflation (ESWD)

1. *Initialize:*

   $j := 0,\ U_0 := \{\emptyset\},\ \Sigma_0 := 0.$

2. *Compute next eigenvector (s) :*

   Call algorithm (A) to compute the eigenvalue λ_{j+1} (resp. the conjugate pair of eigenvalues $\lambda_{j+1}, \lambda_{j+2} \equiv \bar{\lambda}_{j+1}$) of largest real part of the matrix $A_j \equiv A - U_j \Sigma_j U_j^H$, along with an eigenvector y (resp. the real part and imaginary part y_R, y_I of the complex pair of eigenvectors). Choose the next shift σ_{j+1}, and define $\Sigma_{j+1} :=\ Diag\ \{\sigma_1, \sigma_2, \ldots \sigma_{j+1}\}$.

3. *Orthonormalize:*

   Orthonormalize the vector y (resp. the vectors y_R, y_I) against the vectors $u_1, u_2, \ldots u_j$, to get u_{j+1}, (resp. u_{j+1}, u_{j+2}).

   Set $U_{j+1} := [U_j, u_{j+1}],\quad j := j + 1$, (resp. $U_{j+2} := [U_j, u_{j+1}, u_{j+2}],\quad j := j + 2$.)

4. *Test:*

   If $\ j<p\ $ goto 2, else set $p := j$, compute $R_p := U_p^H A U_p$ and exit.

We point out that the above algorithm has as a parameter the algorithm (A) , which delivers the eigenvalue(s) with largest real part(s) with its (their) associated eigenvector(s). The shift σ_{j+1} in step 2, is chosen so that the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_p$ will in turn be the ones with largest real parts during the algorithm. There is much freedom in choosing the shift but it is clear that if it is too large then a poor performance in step 2 of the algorithm will result. Ideally, we might consider choosing σ so the real part of the eigenvalue just computed, i.e. λ_j coincides with that of the last eigenvalue λ_N. This yields $\sigma_{j+1} = Re(\lambda_j - \lambda_N)$. Clearly, this value is not available beforehand but it suffices to have a rough estimate. Practically, we found it convenient and not restrictive in any way to take all shifts equal to some equal value σ determined at the very first step $j = 1$. The matrix Σ_j then becomes σI.

For step 2, we will give more detail in the next sections on how to compute the eigenvectors y or the pair of conjugate eigenvectors $y_R \pm iy_I$. A crucial point here is that the matrix A_j is never formed explicitly, since this would fill the matrix and is highly ineffective. Clearly, if p is large the computational time of each matrix by vector multiplication becomes very expensive. Another potential difficulty which we consider in detail later is the building up of rounding errors.

In step 3, several possibilities of implementation exist. The simplest one which we have adopted in our codes consists in a modified Gram Schmidt algorithm which allows for up to two reorthogonalizations depending on level of cancellation. Another more expensive method of orthogonalizing a set of vectors which is somewhat more robust is the Householder algorithm.

Before exiting in step 4, the upper triangular matrix R_p is computed. For brevity we have omitted to say in the algorithm that one need only compute the upper quasi-triangular part since it is known in theory that the lower part is zero. Note, that the presence of 2 x 2 diagonal blocks requires a particular treatment. Alternatively, we may compute all the elements of the upper Hessenberg part of R_p, at a slightly higher cost. However, as will be seen in the appendix this is not necessarily the best choice. In the presence of round-off, the matrix $R_p \equiv U_p^H A U_p$ is slightly different from the Schur matrix, and computing its eigenvalues correponds to applying a Galerkin process onto the subspace spanned by the block U_p. The appendix deals in detail with the error analysis of the Schur Wielandt deflation.

2.4 Implicit deflation procedures

In many instances explicit deflation can be replaced by a procedure that blends more naturally with the structure of the projection method such as Arnoldi or subspace iteration. The simplest illustration of this technique is with Arnoldi's algorithm the standard version of which is outlined next.

Algorithm: Arnoldi

```
1. Initialize:
    Choose an initial vector $v_1$ of norm unity.
2. Iterate: Do $j = 1, 2, ..., m$
    1. Compute $w := Av_j$
    2. Compute a set of $j$ coefficients $h_{ij}$ so that
           $$w := w - \sum_{i=1}^{j} h_{ij} vi \qquad (26)$$
       is orthogonal to all previous $v_i$'s.
    3. Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$.
```

The above algorithm produces an orthonormal basis of the Krylov subspace

$$K_m = span\{v_1, Av_1, \dots, A^{m-1}v_1\}.$$

The $m \times m$ upper Hessenberg matrix H_m consisting of the coefficients h_{ij} computed by the algorithm represents the restriction of the linear transformation A to the subspace K_m, with respect to this basis, i.e., we have $H_m = V_m^T A V_m$, where $V_m = [v_1, v_2, \dots, v_m]$. Approximations to some of the eigenvalues of A can be obtained from the eigenvalues of H_m. This is Arnoldi's method in its simplest form. In practice m can be fixed and the algorithm can be used iteratively, i.e., it is restarted with v_1 equal to the last approximate eigenvector associated with the desired eigenvalue, until convergence is achieved. This is for the case where only one eigenvalue/ eigenvector pair must be computed. In case several such pairs must be computed, there are two possible options. The first, suggested in [29] is to take v_1 to be a linear combination of the approximate eigenvectors. For example, if we need to compute the p rightmost eigenvectors, we may take $\hat{v}_1 = \sum_{i=1}^{p} \rho_i \tilde{u}_i$, where the eigenvalues are numbered in decreasing order of real parts. The vector v_1 is then obtained from normalizing $\hat{v}_1$. The simplest choice for the coefficients $\rho_i v$ is to take $\rho_i = 1, i = 1, ..p$. There are several drawbacks to this approach, the most important of which being that there is no easy way of choosing the coefficients ρ_i in a systematic manner. The result is that for hard problems, convergence is difficult to achieve.

An alternative is to compute one eigenpair at a time and use deflation. One can use deflation on the matrix A explicitly as was described earlier. Another implementation, which we now describe, is to work with a single basis $v_1, v_2, ..., v_m$ whose first vectors are the Schur vectors that have already computed. Suppose that $k-1$ such vectors have converged and call them $v_1, v_2, ..., v_{k-1}$. Then we choose a vector v_k which is orthogonal to $v_1, ..., v_{k-1}$ and of norm 1. Next we perform $m-k$ steps of an Arnoldi process in which orthogonality of the vector v_j against all previous $v_i's$, including $v_1, ..., v_{k-1}$ is enforced. This generates an orthogonal basis of the subspace

$$span\{v_1, \dots, v_{k-1}, v_k, Av_k, \dots, A^{m-k}v_k\} \qquad (27)$$

Thus, the dimension of this modified Krylov subspace is constant and equal to m in general. A sketch of this implicit deflation procedure applied to Arnoldi's method is the following.

Algorithm: Arnoldi's method with implicit deflation

```
A. Initialize:
    Choose an initial vector $v_1$ of norm unity.
B. Iterate on eigenvalues: Do $k = 1, 2, ..., p$
    1. Arnoldi Iteration. For $j = k, k+1, ..., m$ do:
        • Compute $w := Av_j$
        • Compute a set of $j$ coefficients $h_{ij}$ so that $w := w - \sum_{i=1}^{j} h_{ij} v_i$ is orthogonal to
          all previous $v_i$'s, $i = 1, 2, ..., j$
        • Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$.
    2. Compute approximate eigenvector of $A$ associated the eigenvalue $\tilde{\lambda}_k$ as well as its
       associated residual norm estimate $\rho_k$.
    3. Orthonormalize this eigenvector against all previous $v_j$'s to get the approximate Schur
       vector $\tilde{u}_k$ and define $v_k := \tilde{u}_k$.
    4. If $\rho_j$ is small enough then accept $\tilde{v}_k$ as the next Schur vector, and compute $h_{i,k} =$
       $(Av_k, v_i)$ $i = 1, .., k$. Go to End_Loop_B. Else goto B.1.
End_Loop_B
```

Note that in the B-loop, the Schur vectors associated with the eigenvalues $\lambda_1, ..., \lambda_{k-1}$ are frozen and so is the corresponding upper triangular matrix corresponding to these vectors. As a new Schur vector has converged, step B.4 computes the k-th column of R associated with this new basis vector. In the subsequent steps, the approximate eigenvalues are the eigenvalues of the $m \times m$ Hessenberg matrix H_m defined in the algorithm and whose $k \times k$ principal submatrix is upper triangular For example when $m = 6$ and after the second Schur vector, $k = 2$, has converged, the matrix H_m will have the form,

$$H_m = \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{pmatrix} \qquad (28)$$

Therefore in the subsequent steps, we will consider only the eigenvalues that are not associated with the 2×2 upper triangular matrix.

3 Shift and invert strategies.

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

$$(A - \sigma I)^{-1} \tag{29}$$

for standard problems and with (for example)

$$(K - \sigma M)^{-1} M \tag{30}$$

for a generalized problem of the form $Kx = \lambda Mx$. Strategies for adaptively choosing new shifts and deciding when to shift and factor $(K - \sigma M)$ are usually referred to as Shift-and-Invert strategies. Thus, Shift-and-Invert simply consists of transforming the original problem $(A - \lambda I)x = 0$ into $(A - \sigma I)^{-1}x = \mu x$. The transformed eigenvalues μ_i are usually far better separated than the original ones which results in better convergence in the projection type algorithms. However, there is a trade-off when using Shift-and-Invert, because the original matrix by vector multiplication which is usually inexpensive, is now replaced by the more complex solution of a linear system at every step. When a new shift σ is selected, the LU factorization of the matrix $(A - \sigma I)$ is performed and subsequently, at every step of Arnoldi's algorithm (or any other projection algorithm), an upper and a lower triangular system are solved. Moreover, the cost of the initial factorization can be quite high and in the course of an eigenvalue calculation, one needs to use several shifts, i.e. several factorizations. Despite these additional costs Shift-and-Invert is an extremely useful technique especially for generalized eigenvalue problems.

If the shift σ is suitably chosen the matrix $C = (A - \sigma I)^{-1}$, will have a spectrum with much better separation properties than the original matrix A and therefore should require far less iterations to converge. Thus, the rationale behind shift and invert technique is that factoring the matrix $(A - \sigma I)$ once, or a few times during a whole run in which σ is changed a few times, is a price worth paying because the number of iterations required with C is so much smaller than that required with A that the expense of the factorization is paid off. For the symmetric generalized eigenvalue problem $Mx = \lambda Kx$ there are compelling reasons for employing a Shift-and-Invert technique. These reasons are discussed at length in [22], [23], [10] and [40], the most important one being that since we must factor one of the matrices K or M in any case, there is little incentive in not factoring $(K - \sigma M)$ instead, to gain faster convergence. Shift and invert has now become a fairly standard tool in structural analysis because of the predominance of generalized eigenvalue problems in this applications area.

For nonsymmetric eigenvalue problems, much remains to be done to derive efficient Shift-and-Invert strategies. Parlett and Saad [24] have examined different ways of dealing with the situation where the matrices M and K are real and banded but the shift σ is complex. One such possibility is to replace the complex operator $(K - \sigma M)^{-1}M$ by the real one

$$Re[(K - \sigma M)^{-1} M] \tag{31}$$

whose eigenvectors are the same as those of the original problem and whose eigenvalues μ_i are related to the eigenvalues λ_i of (M, K) by

$$\mu_i = \frac{1}{2}[\frac{1}{\lambda_i - \sigma_i} + \frac{1}{\lambda_i - \bar{\sigma}_i}] \tag{32}$$

One clear advantage of using (31) in place of (30) is that the latter operator is real and therefore all the work done in the projection method can be performed in real arithmetic. A nonnegligible additional benefit is that the complex conjugate pairs of eigenvalues of original problem are also approximated by complex conjugate pairs thus removing some potential difficulties in distinguishing these pairs when they are very close. On the practical side, the matrix $(K - \sigma M)$ must be factored into the product LU of a lower triangular matrix L and an upper triangular matrix U. Then every time the vector $w = Re[(K - \sigma M)^{-1} M]v$ must be computed, the forward and backward solves are processed in the usual way and then the real part of the resulting vector is taken to yield w.

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

Our experimental code ARNINV based on this approach implements a simple strategy which requires two parameters m_1, m_2 from the user and proceeds as follows. The code starts by using σ_0 as an initial shift and calls Arnoldi's algorithm with $(A - \sigma_0 I)^{-1}$ Arnoldi to compute the eigenvalue of A closest to σ_0. Arnoldi's method is used with restarting, i.e., if an eigenvalue fails to converge after the Arnoldi loop we rerun Arnoldi's algorithm with the initial vector replaced by the eigenvalue associated with the eigenvalue closest to σ_0. The strategy for changing the shift is dictated by the second parameter m_2. If after m_2 calls to Arnoldi with the shift σ_0 the eigenpair has not yet converged then the shift σ_0 is changed to the best possible eigenvalue close to σ_0 and we repeat the process. As soon as the eigenvalue has converged we deflate it using Schur deflation as described in the previous section. The algorithm can be summarized as follows.

Algorithm: Shift-and-invert Arnoldi's method with implicit deflation

A. Initialize:

Choose an initial vector v_1 of norm unity, an initial shift σ, the dimension m_1 of the Krylov subspaces to be used, and the number m_2 of calls to Arnoldi before reshifting.

B. Eigenvalue loop: Do $k = 1, 2, ..., p$

1. Compute the LU factorization of $(A - \sigma I)$.
2. If $k > 1$ then (re)-compute $\{h_{ij} = ((A - \sigma I)^{-1} v_j, v_i)\}_{i,j=1,k-1}$.
3. Arnoldi Iteration. For $j = k, k+1, ..., m$ do:
    - Compute $w := (A - \sigma I)^{-1} v_j$
    - Compute a set of j coefficients h_{ij} so that $w := w - \sum_{i=1}^{j} h_{ij} v_i$ is orthogonal to all previous v_i's, $i = 1, 2, ..., j$
    - Compute $h_{j+1,j} := \|w\|_2$ and $v_{j+1} := w/h_{j+1,j}$.
4. Compute eigenvalues of H_m of largest modulus and get corresponding approximate eigenvector of $(A - \sigma I)^{-1}$ and an estimated error ρ_k on the eigenvalue.
5. Orthonormalize this eigenvector against all previous v_j's to get the approximate Schur vector $\tilde{u}_k$ and define $v_k := \tilde{u}_k$;
6. If ρ_k is small enough then accept v_k as the next Schur vector. Go to End_Loop_B.
7. If the number of restarts with the same shift exceeds m_2 select a new shift and goto 1. Else restart Arnoldi's algorithm, i.e., goto 3.

End_Loop_B.

A point of detail in the algorithm is that the $k \times k$ principal submatrix of the Hessenberg matrix H_m is recomputed whenever the shift changes. The reason is that this submatrix represents the matrix $(A - \sigma I)^{-1}$ in the first k Schur vectors and therefore it must be updated as σ changes. This is in contrast with the simple Arnoldi procedure with deflation described earlier. The above algorithm is described for general complex matrix and there is not attempt in it to avoid complex arithmetic in case the original matrix is real. In this situation, we must replace $(A - \sigma I)^{-1} v_j$ in B.2 by $Re[(A - \sigma I)^{-1} v_j]$ and make sure that we select the eigenvalues corresponding to the eigenvalues of A closest to σ. We also need to replace the occurrences of eigenvectors by the pair of real parts and imaginary parts of the eigenvectors.

4 Polynomial Preconditioned Arnoldi Algorithm

There are various ways of preconditioning a linear linear system $Ax = b$ prior to solving it by a Krylov subspace method. Preconditioning consists in transforming the original linear system into one which requires fewer iterations with a given Krylov subspace method, without increasing the cost of each iteration too much. For eigenvalue problems similar methods have not been given much attention although the Shift-and-Invert technique can be viewed as a means of preconditioning. Moreover, Davidson's method is nothing but a form of preconditioned Lanczos algorithm, where the preconditioning matrix is a diagonal matrix which varies at every step. This idea has been exploited by Morgan and Scott [21] who propose a generalization of Davidson's algorithm based on more general preconditioners. Scott [39] also propose a preconditioned Lanczos procedure for the generalized eigenvalue problem. These techniques involve approximating the inverse of $(A - \sigma I)$ by a matrix of the form $(M - \sigma I)^{-1}$, where M can be some approximation of A. Note that in the ideal case where σ is an exact eigenvalue of A, and as $(M-\sigma I)^{-1}$ is usually nonsingular then the rest of the eigenvalues will tend to cluster around one. This good separation will make the algorithm deliver the eigenvalue closest to σ very quickly. These alternatives to Shift-and-Invert might be useful in the case where factoring the matrix $(A - \sigma I)$ is out of the question because of the size of the problem.

For a classical eigenvalue problem, one alternative is to use polynomial preconditioning as is described next. The idea of polynomial preconditioning is to replace the operator B by a matrix of the form,

$$B_r = p_r(A) \tag{33}$$

where p_r is a degree r polynomial. Ruhe [28] considers a more general method in which p_r is not restricted to being a polynomial but can be a rational function. When an Arnoldi type method is applied to B_r, we do not need to form B_r explicitly, since all we will ever need in order to multiply a vector x by the matrix B_r is k matrix-vector products with the original matrix A and some linear combinations.

Instead of attempting to compute several eigenvalues of A at once as was suggested in [29, 32, 31] the method proposed here is to compute only one eigenvalue at a time or possibly a pair of complex conjugate eigenvalues at a time. Deflation is then used to compute the next desired eigenvalues and eigenvectors until satisfied. The goal is to improve robustness, sometimes perhaps at the expense of efficiency. The preconditioning methods rest on the idea that all the difficulties in Arnoldi type methods, come from the poor separation of the desired eigenvalues. The real problem is that often the desired eigenvalues are clustered while the non wanted ones are well separated, which results in the method being unable to retrieve any element of the cluster and leads to very poor performance, often divergence.

For fast convergence, we would ideally like that the next wanted eigenvalue of A be transformed by p_r into an eigenvalue of B_r that is very large as compared with its remaining eigenvalues. There are many ways of providing a satisfactory solution to this problem. The one considered here is derived from [31]. First we impose the constraint

$$p_r(\lambda_1) = 1 \tag{34}$$

Thus, we can attempt to minimize some norm of p_r in some region subject to the constraint (34). We can choose the norm of the polynomials, to be either the infinity norm or the L_2-norm. Because it appears that the L_2-norm offers more flexibility and performs usually slightly better than the infinity norm, we will only consider a technique based on the least squares approach. We should emphasize, however, that a similar technique using Chebyshev polynomials can easily be developed. The procedure for computing the least squares polynomials has been described in detail elsewhere and we will refer the reader to the articles [31, 34].

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

Polynomial Preconditioned Arnoldi with Deflation

A. *Start:* Choose the degree r of the polynomial p_r, the dimension m of the Arnoldi subspaces and an initial vector v_1.

B. *Initial Arnoldi Step:* Using the initial vector v_1, perform m steps of the Arnoldi method with the matrix A.

C. *Eigenvalue loop.* Do $k = 1, 2, ..., p$:

1. *Projection Step:*
   - Obtain the matrix $A_m = V_m^T A V_m$ and its m eigenvalues $\{\tilde{\lambda}_1, \ldots \tilde{\lambda}_m\}$ and eigenvectors $\tilde{y}_i$.
   - Compute the approximate eigenvector $V_m \tilde{y}_k$, and orthogonalize it against all previous v_i's to get the next approximate Schur vector $\tilde{u}_k$. Get the corresponding residual norms ρ_k.
   - If ρ_k is small enough then goto End_loop_C.
   - Adapt: From the previous convex hull and the set $\{\tilde{\lambda}_{k+1}, \ldots, \tilde{\lambda}_m\}$ construct a new convex hull of the unwanted eigenvalues.
   - Compute the new least squares polynomial p_r of degree r.
2. *Arnoldi iteration:*
   Starting with $v_k = \tilde{u}_k$, generate $m - k$ vectors by Arnoldi's method applied to the matrix $B_r = p_r(A)$. The result is a set of m orthonormal vectors $V_m = [v_1, v_2, ..., v_m]$. Go to 1.

End_loop_C

When passing from step 2 to step 3, it is not necessary to actually compute the matrix A_m which is available in step 2 as the Arnoldi matrix H_m. We have skipped some details concerning the deflation process because of the resemblance with the process of the Shift-and-invert algorithm described earlier.

5 Numerical experiments

All numerical tests have been performed on an Alliant FX-4, using double precision, i.e., the unit roundoff is $2^{-56} \approx 1.3877 \text{ x } 10^{-17}$. Our test example, taken from [27], models concentration waves in reaction and transport interaction of some chemical solutions in a tubular reactor. The concentrations $x(\tau, z), y(\tau, z)$ of two reacting and diffusing components, where $0 \leq z \leq 1$ represents a coordinate along the tube, and τ is the time, are modeled by the system: [27]:

$$\frac{\partial x}{\partial \tau} = \frac{D_x}{L^2} \frac{\partial^2 x}{\partial z^2} + f(x, y), \tag{35}$$

$$\frac{\partial y}{\partial \tau} = \frac{D_y}{L^2} \frac{\partial^2 y}{\partial z^2} + g(x, y), \tag{36}$$

with the initial condition

$$x(0, z) = x_0(z), \quad y(0, z) = y_0(z), \quad \forall\ z \in\ [0, 1],$$

and the Dirichlet boundary conditions:

$$x(0, \tau) = x(1, \tau) = \bar{x}$$

$$y(0, \tau) = y(1, \tau) = \bar{y}.$$

The linear stability of the above system is traditionally studied around the steady state solution obtained by setting the partial derivatives of x and y with respect to time to be zero. More precisely, the stability of the system is the same as that of the Jacobian of (35) - (36) evaluated at the steady state solution. In many problems one is primarily interested in the existence of limit cycles, or equivalently the existence of periodic solutions to (35), (36). This translates into the problem of determining whether the Jacobian of (35), (36) evaluated at the steady state solution admits a pair of purely imaginary eigenvalues.

We consider in particular the so-called Brusselator wave model [27] in which

$$f(x, y) = A - (B + 1)x + x^2 y$$

$$g(x, y) = Bx - x^2 y.$$

Then, the above system admits the trivial stationary solution $\bar{x} = A$, $\bar{y} = B/A$. A stable periodic solution to the system exists if the eigenvalues of largest real parts of the Jacobian of the right hand side of (35), (36) is exactly zero. For the purpose of verifying this fact numerically, one first needs to discretize the equations with respect to the variable z and compute the eigenvalues with largest real parts of the resulting discrete Jacobian.

For this example, the exact eigenvalues are known and the problem is analytically solvable. The article [27] considers the following set of parameters

$$D_x = 0.008, \quad D_y = \frac{1}{2} D_x = 0.004,$$

$$A = 2, \quad B = 5.45$$

The bifurcation parameter is L. For small L the Jacobian has only eigenvalues with negative real parts. At $L \approx 0.51302$ a purely imaginary eigenvalue appears. Our tests verify this fact.

Let us discretize the interval $[0, 1]$ using $n + 1$ points, and define the mesh size $h \equiv 1/n$. The discrete vector is of the form $\binom{x}{y}$ where x and y are n-dimensional vectors. Denoting by f_h and g_h the corresponding discretized functions f and g, the Jacobian is a 2 x 2 block matrix in which the diagonal blocks $(1, 1)$ and $(2, 2)$ are the matrices

$$\frac{1}{h^2} \frac{D_x}{L^2} \, Tridiag\{1, -2, 1\} + \frac{\partial f_h(x, y)}{\partial x}$$

and

$$\frac{1}{h^2} \frac{D_y}{L^2} \, Tridiag\{1, -2, 1\} + \frac{\partial g_h(x, y)}{\partial y}$$

respectively, while the blocks $(1, 2)$ and $(2, 1)$ are

$$\frac{\partial f_h(x, y)}{\partial y} \quad \text{and} \quad \frac{\partial g_h(x, y)}{\partial x}$$

respectively. Note that since the two functions f and g do not depend on the variable z, the Jacobians of either f_h or g_h with respect to either x or y are scaled identity matrices. We denote by A the resulting $2n$ x $2n$ Jacobian matrix. We point out that the exact eigenvalues of A are readily computable, since there exists a quadratic relation between the eigenvalues of the matrix A and those of the classical difference matrix $Tridiag\{1, -2, 1\}$.

We will refer to the shift-and-invert Arnoldi algorithm described in Section 3 as ARNINV and to the polynomial preconditioned Arnoldi method of Section 4 as ARNLS. It is difficult to select a suitable stopping criterion for nonsymmetric eigenvalue problems. In our case we have adopted to stop as soon as the residual norm is smaller than some tolerance ϵ. However, the matrices may be scaled differently and we decided to scale the residual norms by the average singular value of the Hessenberg matrix produced by the projection process. More precisely, at every step we compute the square of the Frobenius norm $f_m = \text{Trace } (H_m^H H_m)$, and take as an estimate of the error of the computed pair eigenvalue/eigenvector the number

$$\frac{\rho}{\sqrt{\frac{1}{m} f_m}},$$

where ρ is the computed residual norm provided by the method. Note that the denominator represents the square root of the average of the squares of the singular values of H_m. In ARNLS the same scaling is used except that for the projection step (Step 4 of Algorithm ARNLS), H_m is replaced by the matrix A_m. Recall [29] that it is not necessary to compute the eigenvectors explicitly in Arnoldi in order to get the residual norms because these are equal to the products of $h_{m+1,m}$ by the last component of the corresponding normalized eigenvectors of the matrix H_m.

We used a discretization of $n = 100$ subintervals, i.e., the size of the resulting matrix is 200. We tested ARNINV to compute the six rightmost eigenvalues of A. We took as initial shift the value $\sigma = 0$, and $m_1 = 15$, $m_2 = 10$. In this case ARNINV delivered all the desired eigenvalues by making four calls to the Arnoldi subroutine and there was no need for chaning shifts. The tolerance imposed was $\epsilon = 10^{-7}$. The result of the execution is shown in Figure 1. What is shown in the figure is the progress of the algorithm after each projection (Arnoldi) step. The headings indicate the number of the eigenvalue that is being computed. Thus, when Arnoldi is trying to compute the eigenvalue number 3, it has already computed the first two (in this case a complex conjugate pair), and has deflated them. We print the eigenvalue of interest, i.e., the one we are trying to compute, plus the one that is likely to converge after it. The last column shows the actual residual norm achieved.

We rerun the above test with the initial shift σ, namely $\sigma_0 = -0.5 + 0.2i$ and we changed m_2 to $m_2 = 3$. Initially, the run looked similar to the previous one. A pair of complex conjugate eigenvalues were found in the first Arnoldi iteration, then another pair in the second iteration, then none in the third iteration and one pair in the fourth iteration. It took two more iterations to get the eigenvalues number 7 and 8. For the last eigenvalue a new shift was taken because it took three Arnoldi iterations without success. However the next shift that was taken was already an excellent approximation and the next eigenvalue was computed in the next iteration. The cost was much higher than the previous run with the cpu time climbing to approximately 5.65 seconds.

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

- Total number of matrix by vector multiplications for the run = 2053;
- Number of calls to the projection subroutines = 9;
- Total cpu time used = 3.88 sec.

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

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

Both the number of projection steps and the execution times have been drastically reduced and have come closer to those obtained with shift-and-invert. One of the disadvantages of polynomial preconditionings is precisely this wide variation in performance depending on the choice of the parameters. To some extent there is a similar dependence of the performance of ARNINV on the initial shift, although in practice a good initial shift is often known. A superior feature of shift and invert is that it allows to compute eigenvalues inside the spectrum. Polynomial preconditioning can be generalized to this case but does not perform too well in general. We should also comment on the usefulness of using polynomial preconditioning in general. One often heard argument against polynomial preconditioning is that is it suboptimal: in the symmetric case the conjugate gradient and the Lanczos methods are optimal polynomial processes in that they provide the best possible approximation, in some sense, to the original problem from Krylov subspaces. Hence the argument that polynomial preconditioning would

```
-------------------------------------------------------------
     computing eigenvalue number   1
-------------------------------------------------------------
    re(lambda)                  im(lambda)     *  res. norm *
 0.1807540453D-04 + i.      0.2139497548D+01  *  0.212D-09 *
 0.1807540453D-04 + i.     -0.2139497548D+01  *  0.212D-09 *
-0.6747097569D+00 + i.      0.2528559918D+01  *  0.224D-06 *
-0.6747097569D+00 + i.     -0.2528559918D+01  *  0.224D-06 *
-------------------------------------------------------------
     computing eigenvalue number   3
-------------------------------------------------------------
    re(lambda)                  im(lambda)     *  res. norm *
-0.6747097569D+00 + i.      0.2528559918D+01  *  0.479D-13 *
-0.6747097569D+00 + i.     -0.2528559918D+01  *  0.479D-13 *
-0.2780085122D+01 + i.      0.2960250300D+01  *  0.336D-01 *
-0.2780085122D+01 + i.     -0.2960250300D+01  *  0.336D-01 *
-------------------------------------------------------------
    computing eigenvalue number   5
-------------------------------------------------------------
  re(lambda)                  im(lambda)     *  res. norm *
-0.1798530837D+01 + i.      0.3032164644D+01  *  0.190D-06 *
-0.1798530837D+01 + i.     -0.3032164644D+01  *  0.190D-06 *
-------------------------------------------------------------
     computing eigenvalue number   5
-------------------------------------------------------------
    re(lambda)                  im(lambda)     *  res. norm *
-0.1798530837D+01 + i.      0.3032164644D+01  *  0.102D-11 *
-0.1798530837D+01 + i.     -0.3032164644D+01  *  0.102D-11 *
-0.2119505960D+02 + i.      0.1025421954D+00  *  0.749D-03 *
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 average error =     0.6820322E-14
 total execution time = 2.13 sec.
```

Figure 1: Convergence history of ARNINV for first test. Each separate output corresponds to a call to Arnoldi's module

not do as well if one counts the total number of matrix by vector multiplications. The counter argument in the nonsymmetric case is a simple one: the optimality result is no longer true. For linear systems, algorithms such as GMRES or GCR, are known to be optimal in the sense that they obtain the solution with smallest residual norm in the Krylov subspace [36, 35]. However, this optimality is only valid for the highly impractical case where a full orthogonalization process with no restarting or truncation is undertaken at every step. In fact even in the symmetric case the optimality result is only true in exact arithmetic, which is far from the real situation where loss of orthogonality is a rather severe and damaging phenomenon.

The next question is whether a simple restarted Arnoldi algorithm would perform better than a polynomial preconditioned method. The answer is a definite no. A run with ARNIT [33] an iterative Arnoldi method with deflation failed even to deliver the first eigenvalue of the test matrix used in the above example. The initial vector was the same and we tried two cases $m = 15$, which did not show any sign of convergence and $m = 20$ which might have eventually converged but was extremely slow.

6 Summary and Conclusion

We have presented essentially two methods for computing a few eigenvalues and the corresponding eigenvectors or Schur vectors of large nonsymmetric matrices. Both techniques rely heavily on a Schur Wielandt deflation procedure and preconditioning. The first one uses Shift-and-Invert preconditioning and the second a form of polynomial preconditioning. These methods are of interest only when the number of eigenvalues to be computed is relatively small, such as when dealing with the stability analysis in nonlinear differential equations, or in the analysis of various bifurcation phenomena. Our analysis of the appendix and our experiments indicates that the Schur-Wielandt deflation is safe to use in general. There is an a-posteriori upper bound (see appendix) which can be used in practice to estimate the accuracy of the computed basis of the invariant subspace.

Although not mentioned before, the deflation technique can also be of great help when dealing with the generalized eigenvalue problem. If one uses an Arnoldi or a nonsymmetric Lanczos method, big savings in computational cost can be achieved with deflation because it allows one to compute more eigenvalues with the same shift and so fewer expensive factorizations must be performed. In essence the selective orthogonalization technique developed by Parlett and Scott [23, 38] realizes a similar deflation technique in the symmetric case but does so in a more economical way.

7 APPENDIX: Error Analysis of Schur Wielandt deflation

In this appendix, we propose a few a posteriori error bounds in order to analyse the stability of the deflation technique. Typically, at each step $j = 1, 2, \dots p$ of the deflation process we compute an approximate eigenvalue λ_j and an associated normalized eigenvector y_j of the matrix $A_{j-1} \equiv A - U_{j-1}\Sigma_{j-1}U_{j-1}^H$. As a convention we define A_0 to be the matrix A. The approximate eigenpair satisfies the relation

$$A_{j-1}y_j = \lambda_j y_j + \eta_j, \quad j = 1, \dots p \tag{37}$$

where the residual vector η_j is some vector of small norm and is assumed to include both the effects of approximation and rounding. It is assumed that the matrix U_p is orthonormal to working precision. Our purpose is to provide some information on the accuracy of the Schur basis U_p and possibly of the eigenvalues obtained from the approximate eigenvalues $\lambda_j, j = 1, \dots p$.

At step number j, the vector y_j is orthogonalized against $u_1, u_2, \dots, u_{j-1}$ to obtain the j^{th} approximate Schur vector u_j. This is realized by a Gram-Schmidt process and as a result the following relationship between the vectors u_i and y_j holds:

$$\sum_{i=1}^{j} \beta_{i,j} u_i = y_j \quad j = 1, 2, \dots p. \tag{38}$$

Denoting by b_j the vector of p components $\beta_{1,j}, \beta_{2,j}, \dots, \beta_{jj}, 0, 0, \dots 0$, the above relation can be rewritten as

$$U_p b_j = y_j. \tag{39}$$

Replacing this relation in (37), we have

$$(A - U_{j-1}\Sigma_{j-1}U_{j-1}^H)U_p b_j = \lambda_j U_p b_j + \eta_j \tag{40}$$

or

$$AU_p b_j = U_{j-1}\Sigma_{j-1}U_{j-1}^H U_p b_j + \lambda_j U_p b_j + \eta_j. \tag{41}$$

Although there are only $p-1$ shifts σ_i used when p eigenvalues are computed, it is convenient to define $\sigma_p \equiv 0$ and

$$\Sigma_p \equiv \; Diag\ \{\sigma_1, \sigma_2, \dots, \sigma_{p-1}, \sigma_p\}. \tag{42}$$

Then (41) becomes

$$AU_p b_j = U_p \left[\Sigma_p + (\lambda_j - \sigma_j)I\right] b_j + \eta_j, \quad j = 1, 2, \dots, p. \tag{43}$$

Let B_p be the p x p upper triangular matrix having as its column vectors the $b_i's$, E_p the N x p matrix having as its column vectors the $\eta_i's$ and $\Lambda_p \equiv Diag\{\lambda_1, \lambda_2, \dots \lambda_p\}$. Then the above relation translates into the matrix relation:

$$AU_p B_p = U_p \left[\Sigma_p B_p + B_p(\Lambda_p - \Sigma_p)\right] + E_p, \tag{44}$$

which we rewrite in a final form as

$$AU_p = U_p \left[\Sigma_p + B_p(\Lambda_p - \Sigma_p)B_p^{-1}\right] + E_p B_p^{-1}. \tag{45}$$

For convenience, we define

$$Z_p \equiv E_p B_p^{-1} \tag{46}$$

and

$$C_p \equiv \Sigma_p + B_p(\Lambda_p - \Sigma_p)B_p^{-1}. \tag{47}$$

Observe that when $\sigma_i = \lambda_i, i = 1, \dots p-1$ then the matrix U_p diagonalizes partially the matrix A if $E_p = 0$.

At the final stage of Algorithm ESWD, there are two ways of post processing before exiting.

- Either one accepts the values $\lambda_i, i = 1, \dots p$ as approximate eigenvalues and does not attempt to improve them. The representation of the section of A in the approximate invariant subspace U_p is taken to be the matrix C_p defined by (47).
- Or one performs a final Galerkin projection onto the subspace spanned by U_p in order to improve the current approximations. This is done by replacing the approximate eigenvalues $\lambda_i, i = 1, \dots p$ by the eigenvalues of the matrix $R_p \equiv U_p^H A U_p$.

We will mainly focus our attention on the second approach, which is more attractive. In this case the Galerkin process involves some extra work, since the computation of the matrix R_p itself costs us p^2 inner products. However, since p is small this is negligible as compared with the total work incurred during the whole computation. Note that R_p is a full matrix with small lower triangular part, and one might still want the partial Schur form corresponding to the improved eigenvalues. This is easily done by computing the Schur factorization of the matrix R_p, $R_p = Q_p S_p Q_p^H$ and then defining the new U_p matrix by $U_{p,new} = U_p Q_p$.

Consider any N x $(N-p)$ matrix $W \equiv [w_1, w_2, \dots w_{N-p}]$ which complements the matrix U_p into an orthonormal N x N matrix, i.e., so that the matrix $[U_p, W]$ is orthonormal. The matrix representation of the matrix A in this new basis is such that

$$A[U_p, W] = [U_p, W] \begin{pmatrix} R_p & X_{12} \\ W^H Z_p & X_{22} \end{pmatrix}, \tag{48}$$

in which $X_{12} = U_p^H A W, X_{22} = W^H A W$, and Z_p, R_p have been defined above.

The above equation indicates that $[U_p, W]$ almost realizes a Schur factorization of A when Z_p is small. In fact, the factorization can be rewritten in the following form:

$$A - [U_p, W] \begin{pmatrix} O & O \\ W^H Z_p & O \end{pmatrix} [U_p, W]^H = [U_p, W] \begin{pmatrix} R_p & X_{12} \\ O & X_{22} \end{pmatrix} [U_p, W]^H. \tag{49}$$

When a Galerkin correction step is taken, then the approximate Schur factorization corresponds to taking U_p as the basis of the eigenspace and R_p as the representation of A in that subspace. As a consequence, in the approach using a correction step, equation (49) establishes that the final result is equivalent to perturbing the initial matrix A by a matrix which is unitarily similar to the matrix

$$\begin{pmatrix} O & O \\ W^H Z_p & O \end{pmatrix}. \tag{50}$$

Thus, the eigenvalues of R_p will be good approximations of those of A if they are well conditioned, whenever the norm of $W^H Z_p$ is small. The first case (no correction) can be treated in the same way and one can easily prove that the perturbation matrix is unitarily similar to

$$\begin{pmatrix} U_p^H Z_p & O \\ W^H Z_p & O \end{pmatrix}. \tag{51}$$

This analysis proves that the key factor for the stability of the deflation method is the way in which the norm of Z_p increases.

We now wish to provide a result which establishes an a-posteriori upper bound of the Frobenius norm of Z_j as j increases. The column vectors $z_j, j = 1, 2, \ldots p$ of Z_p satisfy the relation:

$$\eta_j = \sum_{i=1}^{j} \beta_{ij} z_i \tag{52}$$

from which we derive the upper bound

$$\beta_{jj} ||z_j|| \leq ||\eta_j|| + \sum_{i=1}^{j-1} \beta_{ij} ||z_i||. \tag{53}$$

Using the Cauchy-Schwartz inequality for the last term on the right-hand side we get

$$\beta_{jj} ||z_j|| \leq ||\eta_j|| + \left[\sum_{i=1}^{j-1} \beta_{ij}^2 \right]^{1/2} \cdot \left[\sum_{i=1}^{j-1} ||z_i||^2 \right]^{1/2} . \tag{54}$$

Since we have assumed that the eigenvector y_i, which is orthogonalized against the previous $u_i's$, is of norm unity, an important observation is that the sum of the squares of the β_{ij} is one and β_{jj} represents simply the sine of the angle θ_j between y_j and the subspace spanned by the vectors $u_i, i = 1, \ldots j-1$. Therefore, denoting by ρ_i the Frobenius norm of the matrices $Z_i, i = 1, \ldots p$ the above inequality reads

$$sin(\theta_j) ||z_j|| \leq ||\eta_j|| + cos(\theta_j) \rho_{j-1}. \tag{55}$$

Adding the term $sin(\theta_j)\rho_{j-1}$ to both sides and using the inequality $(a^2 + b^2)^{1/2} \leq a + b$ for the resulting left hand-side we obtain

$$sin(\theta_j) \rho_j \leq ||\eta_j|| + (sin\theta_j + cos\theta_j) \rho_{j-1}, \tag{56}$$

which is restated in the following proposition.

Proposition 7.1 *The Frobenius norms ρ_j of the matrices $Z_j, j = 1, \ldots p$ satisfy the recurrence relation*

$$\rho_j \leq (1 + cot\ \theta_j) \rho_{j-1} + \frac{||\eta_j||}{sin\theta_j}, \tag{57}$$

where θ_j is the acute angle between the eigenvector y_j obtained at the j^{th} deflation step and the previous approximate invariant subspace $span\{U_{j-1}\}$ and where η_j is its residual vector.

It is important to note that since by definition $sin\ \theta_j = \beta_{jj}$ all the quantities involved in the proposition are available during the computation and so the above recurrence is easily computable starting with the initial value $\rho_0 = 0$. The result can be interpreted as follows: if the angle between the computed eigenvector and the previous invariant subspace is small at every step then the process may quickly become unstable. On the other hand if this is not the case then the process is quite safe, for small p. The interesting point is that the above recurrence can practically be used to determine whether or not there is such a risk of instability. The cause of the potential instability is even narrowed down to the orthogonalization process. If each newly

j	$\|\|Z_j\|\|_2$	Upper bound ρ_j
2	0.2679108E-05	0.1161542E-04
4	0.8961249E-05	0.1857656E-04
6	0.1313459E-04	0.3268575E-04
8	0.1398945E-04	0.6703071E-04
10	0.1397979E-04	0.6776894E-04

Table 1: Comparison of the estimated Frobenius norms of the errors in the invariant subspaces with the actual norms.

computed vector y_j were orthogonal to the previous ones then clearly B_p would be the identity matrix and there would be no risk of amplification of errors. This opens up an interesting possibility. Assume that instead of computing an approximate eigenpair λ_j, y_j satisfying the relation (37) one is able by some hypothetical procedure to compute a Schur pair directly, i.e., a pair λ_j, u_j satifying the analogous relation

$$A_{j-1}u_j = \lambda_j u_j + \sum_{i=1}^{j-1} \gamma_{ij} u_i \quad + \eta_j. \tag{58}$$

Then an analysis similar to the one used to establish (45) would easily lead to the relation $AU_p = U_p\tilde{R}_p + E_p$ where $\tilde{R}_p$ is the upper triangular matrix having the diagonal elements $\lambda_i, i = 1, p$ and the off diagonal elements γ_{ij}, while E_p is defined as before. Thus, in this case Z_p is simply replaced by E_p and the process is always stable. In a way, however, the difficulty is rejected to the hypothetical procedure that would compute the Schur pair. As an example, a naive algorithm for computing a Schur pair would be to compute the eigenpair and then orthogonalize the eigenvector y_j against the previous $u_i's$ to get u_j. By doing so a relation of the form (58) is always satisfied and η_j and its norm can be explicitly computed. If $||\eta_j||$ is not sufficiently small one goes back to compute the eigenpair λ_j, y_j to higher accuracy until $||\eta_j||$ is as small as wanted. The issue of whether there may exist other methods that deliver directly Schur vectors, is worth investigating.

In this illustrative test taken from [33], we verify the error bound (57). The test matrix is the same as that of the numerical experiments section, but of size N=100, which corresponds to a discretization of $n = 50$ interior mesh points. We have computed the 10 rightmost eigenvalues and their associated Schur vectors by using an algorithm based on a polynomial accelerated Arnoldi method as described in [33] which is different from ARNLS. We used with $m = 10$, and polynomial of degree 100 = 5 x 20. Here, the stopping criterion for each eigenpair is that the actual residual norm be less than $\epsilon \equiv 10^{-5}$. In other words the norms of the vectors η_i as defined by (37) are less than ϵ except for rounding in the actual computation of this residual which is negligible in view of the fact that ϵ is large compared to the unit round-off. As soon as a new pair of complex conjugate eigenvalues converged, we computed the corresponding new Frobenius norm of Z_j and the corresponding estimate given by (57). The results are shown in Table 1. The 10 rightmost eigenvalues are all complex and so they appear in pairs. In this example, in fact in all our tests conducted with this class of test matrices, there is a good agreement between the estimated norm and its actual value.

References

[1] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] F. L. Bauer. Das verfahren der treppeniteration und verwandte verfahren zur losung algebraischer eigenwertprobleme. *ZAMP*, 8:214–235, 1957.

[3] P. N. Brown and A. C. Hindmarsh. Matrix-free methods for stiff systems of odes. *SIAM J. Num. Anal.*, 23:610–638, 1986.

[4] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comp.*, 11:450–481, 1990.

[5] E. Carnoy and M. Geradin. On the practical use of the Lanczos algorithm in finite element applications to vibration and bifurcation problems. In Axel Ruhe, editor, *Proceedings of the Conference on Matrix Pencils, Lulea, Sweden, March 1982*, pages 156–176, New York, 1982. University of Umea, Springer Verlag.

[6] T. F. Chan and K. R. Jackson. Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms. *SIAM J. Stat. Scien. Comput.*, 7:533–542, 1984.

[7] M. Clint and A. Jennings. The evaluation of eigenvalues and eigenvectors of real symmetric matrices by simultaneous iteration method. *J. Inst. Math. Appl.*, 8:111–121, 1971.

[8] J. Cullum and R. Willoughby. A Lanczos procedure for the modal analysis of very large nonsymmetric matrices. In *Proceedings of the 23rd Conference on Decision and Control, Las Vegas*, 1984.

[9] J. Cullum and R. A. Willoughby. *Large Scale Eigenvalue Problems.* North-Holland, 1986. Mathematics Studies series, Number 127.

[10] T. Ericsson and A. Ruhe. The spectral transformation Lanczos method in the numerical solution of large sparse generalized symmetric eigenvalue problems. *Math. Comp.*, 35:1251–1268, 1980.

[11] L. E. Eriksson and A. Rizzi. Analysis by computer of the convergence of discrete approximations to the euler equations. In *Proceedings of the 1983 AIAA conference, Denver 1983*, pages 407–442, Denver, 1983. AIAA.

[12] C. W. Gear and Y. Saad. Iterative solution of linear equations in ode codes. *SIAM J. Sci. Statist. Comput.*, 4:583–601, 1983.

[13] G. H. Golub and C. Van Loan. *Matrix Computations.* Academic Press, New York, 1981.

[14] H. Hlavacek and H. Hofmann. Modeling of chemical reactors XVI. *Chemical Eng. Sci.*, 25:1517–1526, 1970.

[15] A. Jennings and W. J. Stewart. Simultaneous iteration for partial eigensolution of real matrices. *J. Math. Inst. Appl.*, 15:351–361, 1980.

[16] A. Jennings and W. J. Stewart. A simultaneous iteration algorithm for real matrices. *ACM, Trans. of Math. Software*, 7:184–198, 1981.

[17] A. Jepson. *Numerical Hopf Bifurcation.* PhD thesis, Cal. Inst. Tech., Pasadena, CA., 1982.

[18] D. D. Joseph and D. H. Sattinger. Bifurcating time periodic solutions and their stability. *Arch. Rat. Mech. Anal,*, 45:79–109, 1972.

[19] T. Kerkhoven and Y. Saad. Acceleration techniques for decoupling algorithms in semiconductor simulation. Technical Report 684, University of Illinois, CSRD, Urbana, IL., 1987.

[20] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. of Standards*, 45:255–282, 1950.

[21] R. B. Morgan and D. S. Scott. Generalizations of davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Statist. Comput.*, 7:817–825, 1986.

[22] B. N. Parlett. How to solve $(K - \lambda M)z = 0$ for large K and M. In E. Asbi et al., editor, *Proceedings of the 2nd International Congress on Numerical Methods for Engineering (GAMNI 2)*, pages 97–106, Paris, 1980. Dunod.

[23] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, 1980.

[24] B. N. Parlett and Y. Saad. Complex shift and invert strategies for real matrices. *Linear Algebra and its Applications*, 88/89:575–595, 1987.

[25] B. N. Parlett and D. Taylor. A look ahead Lanczos algorithm for unsymmetric matrices. Technical Report PAM-43, Center for Pure and Applied Mathematics, Berkeley, California, 1981.

[26] B. N. Parlett, D. R. Taylor, and Z. S. Liu. A look-ahead Lanczos algorithm for nonsymmetric matrices. *Mathematics of Computation*, 44:105–124, 1985.

[27] P. Raschman, M. Kubicek, and M. Maros. Waves in distributed chemical systems: experiments and computations. In P. J. Holmes, editor, *New Approaches to Nonlinear Problems in Dynamics - Proceedings of the Asilomar Conference Ground, Pacific Grove, California 1979*, pages 271–288. The Engineering Foundation, SIAM, 1980.

[28] A. Ruhe. Rational Krylov sequence methods for eigenvalue computations. *Linear Algebra Appl.*, 58:391–405, 1984.

[29] Y. Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.*, 34:269–295, 1980.

[30] Y. Saad. The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems. *SIAM J. Numer. Anal.*, 19:470–484, 1982.

[31] Y. Saad. Least squares polynomials in the complex plane with applications to solving sparse nonsymmetric matrix problems. Technical Report 276, Yale University, Computer Science Dept., New Haven, Connecticut, 1983.

[32] Y. Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Mathematics of Computation*, 42:567–588, 1984.

[33] Y. Saad. Partial eigensolutions of large nonsymmetric matrices. Technical Report YALEU/DCS/RR-397, Yale University, New Haven, CT., 1985.

[34] Y. Saad. Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems. *SIAM J. Num. Anal.*, 24:155–169, 1987.

[35] Y. Saad and M. H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Mathematics of Computation*, 44(170):417–424, 1985.

[36] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[37] G. Sander, C. Bon, and M. Geradin. Finite element analysis of supersonic panel flutter. *Internat. J. Num. Meth. Engng.*, 7:379–394, 1973.

[38] D. S. Scott. *Analysis of the symmetric Lanczos process.* PhD thesis, University of California at Berkeley, Berkeley, CA., 1978.

[39] D. S. Scott. Solving sparse symmetric generalized eigenvalue problems without factorization. *SIAM J. Num. Anal.*, 18:102–110, 1981.

[40] D. S. Scott. The advantages of inverted operators in Rayleigh-Ritz approximations. *SIAM J. on Sci. and Statist. Comput.*, 3:68–75, 1982.

[41] G. W. Stewart. Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices. *Numer. Math.*, 25:123–136, 1976.

[42] G. W. Stewart. SRRIT - a FORTRAN subroutine to calculate the dominant invariant subspaces of a real matrix. Technical Report TR-514, University of Maryland, College Park, MD, 1978.

[43] D. Taylor. *Analysis of the look-ahead Lanczos algorithm.* PhD thesis, Department of Computer Science, Berkeley,CA 94720, 1983.

[44] L. B. Wigton, D. P. Yu, and N. J. Young. GMRES acceleration of computational fluid dynamics codes. In *Proceedings of the 1985 AIAA conference, Denver 1985*, Denver, 1985. AIAA.

[45] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, 1965.