Abstract. We propose several methods based on combinations of deflation techniques and polynomial iteration methods, for computing small invariant subspaces of large matrices, associated with the eigenvalues with largest (or smallest) real parts. We consider both Chebyshev polynomials and least-squares polynomials for the acceleration scheme and we propose a deflation technique which is a variant of Wielandt's deflation that does not require the left eigenvectors of the matrix. As an application we compare our methods on an example issued from a bifurcation problem and show their efficiency when the number of eigenvalues required is small.

Partial Eigensolutions of Large Nonsymmetric Matrices

Youcef Saad

Research Report YALEU/DCS/RR-397
June 1985

This work was supported in part by by ONR grant N00014-82-K-0184 and in part by the Dept. Of Energy under Grant AC02-81ER10996, and in part by Army Research Office under contract DAAG-83-0177.

1. Introduction

1.1. Previous work on solving large nonsymmetric eigenvalue problems

Many important problems in engineering require the computation of a small number of eigenvalues with algebraically largest (or smallest) real parts of a large nonsymmetric real matrix A. Of the few typical examples reported in [26] we only mention the important class of bifurcation problems [12], from which we will draw our main test example. From the numerical point of view, nonsymmetric eigenvalue problems can be substancially more difficult to deal with than the symmetric ones. Perhaps this is one reason for the lack of significant progress on procedures for treating nonsymmetric matrix eigenproblems.

There have been mainly three basic methods for solving large nonsymmetric eigenvalue problems investigated so far. The first is Bauer's subspace iteration method and its many variations [2, 6, 11, 32, 33, 35]. An important drawback of this method, recognized both in the symmetric case [17, 18], and the nonsymmetric case [27, 26] is that it may be exceedingly slow to converge. Another weakness of the subspace iteration method 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. This difficulty, however, can be obviated by using Chebyshev acceleration as is suggested in [26]. The second method is due to Arnoldi [1, 27] 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 excessive as the number of steps increases. Variations on the basic scheme have been proposed [27], which lead to oblique projection type techniques [28], but their theory is not well understood and we will not consider them here.

The third method is the nonsymmetric Lanczos method [7, 14, 19, 20, 34] which is another generalization of the symmetric Lanczos algorithm due originally to Lanczos himself. It produces a tridiagonal matrix whose eigenvalues 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 [28]. Parlett, Taylor and Liu [20] have suggested an elegant solution to the problem of breakdown which has given a bad reputation to the Lanczos method in the past [35]. Cullum and Willoughby [7] extend their symmetric Lanczos algorithm without reorthogonalization, to the nonsymmetric case and suggest a new method for dealing with the resulting non-hermitian tridiagonal matrices. On the whole the main difficulty with the Lanczos method is theoretical, as the method is not quite well understood yet.

To these three basic methods we should add the important shift and invert technique which is not an algorithm in itself but simply consists of transforming the original problem $(A - \lambda I)x = 0$ into $(A - \sigma I)^{-1}x = \mu x$ which may be easier to solve if the shift is close to some eigenvalue of A. Notice that there is a trade-off when using shift and invert, because the basic matrix by vector multiplication which is usually inexpensive, is now replaced by the more complex solution of a linear system at every step: the factorization of the matrix $(A - \sigma I)$ is performed only once and then at every step of any of the above methods, one solves two triangular systems. The cost of the factorization can be quite high and in the course of an eigenvalue calculation, one needs to use several shifts, i.e. several factorizations. Shift and invert has now become a fairly standard tool in structural analysis because there one needs to solve the symmetric generalized eigenvalue problem $Mx = \lambda Kx$ and since there is at least one factorization to perform anyway, shift and invert no longer appears expensive [23, 17, 18, 31].

Comparing the limitations of these methods, we should emphasize that subspace iteration is only able to compute a small number of eigenvalues and associated eigenvectors. To some extent Arnoldi's method presents the same limitations in practice. The nonsymmetric Lanczos algorithm

(especially 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 [7, 19]. On the other hand the Lanczos algorithm requires the use of both the matrix A and of its transpose. As will be seen next, there are applications where 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 often cannot even be approximated with finite differencing.

1.2. Motivation

In this paper we are concerned only with the problem of computing a very small number of eigenvalues and their associated eigenvectors or rather their associated invariant subspace. Our motivation is that in most realistic applications the demand is to compute a very small number of eigenvalues of A of (algebraically) largest real parts or smallest real parts. In these applications one wishes to determine whether a certain system governed by a partial differential equation of the form

$$\frac{du}{dt} = F(u, \theta) \tag{1.1}$$

where F is a partial differential operator, and θ some real parameter, is stable for some value of the parameter θ. Such a system is said to be stable if all the eigenvalues of the Jacobian of F with respect to u, have negative real parts. Hence, all that may be wanted here is to compute one or two (i.e. a comple pair of) eigenvalues. In most bifurcation problems, one is interested in a singular phenomenon occuring past a few singular points, turning points or bifurcation points, but their number seldom exceeds 3 or 4. In other words the number of eigenvalues to compute, i.e. those that have nonnegative real parts is, say, at most 4 real eigenvalues or 4 complex conjugate pairs.

An important observation is that the Jacobian matrix is often not needed explicitly when an eigenvalue algorithm that uses the matrix A only through the matrix vector multiplications $y = Ax$ is employed. This is because the multiplication of the Jacobian J, evaluated at the coordinate u, times a vector x can be performed at low cost with the help of the difference formula

$$Jx \approx \frac{F(u + \epsilon x, \theta) - F(u, \theta)}{\epsilon}, \tag{1.2}$$

where ϵ is some small and carefully chosen scalar.

The approximation (1.2) has been the main instrument in the success of the so-called matrix-free Ordinary Differential Equations solvers [3, 5, 9]: the Jacobian is never computed explicitly which results in significant savings both in computing time and storage. A similar principle has also been employed by Eriksson and Rizzi [8] to compute eigenvalues of various semi-discrete operators used in compressible fluid flow calculations. On the other hand, if the eigenvalue procedure requires the use of A^T, it is not clear how one can avoid the explicit computation of the Jacobian, since the transpose of the Jacobian is not easily approximated in a similar way. We should add however, that this is only valid when the function $F(u, \theta)$ is some complicated nonlinear function for which derivatives are particularly hard to calculate. Such examples abound in scientific applications.

1.3. Overview of results

For the above class of problems a combination of polynomial iteration, such as Chebyshev iteration, with some deflation techniques are quite appropriate. This paper introduces mainly two methods based on this combination:

- A deflation method which is a particular case of Wielandt deflation. An error analysis of the deflation technique is proposed. Polynomial iteration is used to provide a good initial vector for the Arnoldi process. The deflation technique enables us to compute one eigenvalue or a pair of complex conjugate eigenvalues at a time.

- A polynomial preconditioning technique consisting of iterating with the matrix $p(A)$, where p is a polynomial, chosen so that its eigenvalue distribution leads to much faster convergence than would be the case with the original matrix A.

Instead of attempting to compute several eigenvalues of A at once as was suggested in [27, 26, 25] the class of methods propsed in section 2, consists of computing 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. Our goal is to improve robustness, sometimes perhaps at the expense of efficiency. The possible non-availability of the transpose of A as in the above applications, dictates that we choose the deflation technique to be a Wielandt-type deflation which does not require left eigenvectors. We will show a particular type of Wielandt deflation which is naturally suited for computing partial Schur forms.

The preconditioning method proposed in Section 4, rests 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. The usual polynomial acceleration methods consist of starting the Arnoldi iteration with a good initial vector which is computed from a polynomial iteration of the form $z_k = p(A)z_0$, where p is an appropriately chosen polynomial. However, in some cases the eigenvalue separation can be so poor that the Arnoldi process seems even unable to take advantage of a good initial vector and quickly introduces unwanted components. Our idea of the polynonial preconditioned Arnoldi method is to use the polynomial acceleration differently, by simply employing the polynomial iteration as an inner loop for the Arnoldi process. In other words the matrix A is replaced by the preconditioned matrix $p(A)$, whose eigenvalue separation around the desired eigenvalue is much better than that of A.

In the numerical experiments section we consider an example which is a parameter dependent problem of the sort described in section 1.2. Problems of that sort are numerous in structural engineering [4], in aerodynamics (the panel flutter problem [29]), chemical engineering [10], fluid mechanics [13] and many other fields. Our goal is to demonstrate how the proposed methods perform on matrices arising from a typical bifurcation problem.

2. A Schur-Wielandt deflation technique

In the nonsymmetric case most deflation techniques require the knowledge of right and left eigenvectors. However, these deflation procedures of which an example is Hotelling's deflation, can be ill-conditioned because determining eigenvectors of a matrix can be itself an ill-conditioned problem. In fact in the defective case there is no basis of the invariant subspace consisting of eigenvectors and therefore any numerical method that attempts to determine such a basis will likely be ill-conditioned. As suggested by Stewart [33] 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

$$AQ = QR$$

where Q is an N x p complex unitary matrix and R is upper triangular complex matrix. 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 for a given order the factorization is unique in the usual sense of QR factorizations, i.e. the columns of Q are uniquely determined up to a sign of the form $e^{i\theta}$. Thus, whenever we choose an 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.

In this section we describe a deflation technique which is a simple variation of Wielandt's deflation and show that it is very suitable for computing orthonormal bases of invariant subspaces and the corresponding partial Schur forms. We start our discussion with a one vector deflation and then we will generalize the technique to several vectors. In the following we denote by $||.||$ the 2-norm in $\mathbf{C}^N$ and by X^H the transpose of the complex conjugate of a matrix X. 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. Deflation with one vector

Suppose that we have computed the eigenvalue λ_1 of largest real part and its corresponding eigenvector u_1 by some simple algorithm, say algorithm $\mathcal{A}$, which always delivers the eigenvalue of largest real part of the input matrix, along with an eigenvector. For example, in the particular case where all the eigenvalues of A are real and positive Algorithm $\mathcal{A}$ can simply be the power method. In this section we consider the simple case where λ_1 is real. It is assumed that the vector u_1 is normailized so that $||u_1|| = 1$. 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 $\mathcal{A}$ can now be applied to the new matrix to retrieve the pair λ_2, u_2.

Unlike many other deflation techniques, Wielandt's deflation requires only the knowledge of the right eigenvector. The deflated matrix is of the form

$$A_1 = A - \sigma u_1 x^H, \tag{2.1}$$

where x is an arbitrary vector such that $x^H u_1 = 1$, and σ is an appropriate shift. It can be shown that the eigenvalues of A_1 are the same as those of A except for the eigenvalue λ_1 which is transformed into the eigenvalue $\lambda_1 - \sigma$, see [35].

The particular choice $x = u_1$ has the interesting property of preserving the Schur vectors of A. More precisely we can state the following proposition.

Proposition 2.1. *Let u_1 be an eigenvector of A of norm 1, associated with the real eigenvalue λ_1 and let*

$$A_1 \equiv A - \sigma u_1 u_1^H. \tag{2.2}$$

Then the eigenvalues of A_1 are $\lambda_1' = \lambda_1 - \sigma$ and $\lambda_j' = \lambda_j, j = 2, 3 \ldots N$. Moreover, the Schur vectors associated with $\lambda_j', j = 1, 2, 3 \ldots N$ are identical with those of A.

Proof. Let

$$AU = UR \tag{2.3}$$

be the Schur factorization of A, where R is upper triangular and U is orthonormal. Then we have

$$A_1 U = [A - \sigma u_1 u_1^H]U = UR - \sigma u_1 e_1^H = U[R - \sigma_j e_1 e_1^H]$$

The result follows immediately.

∎

2.2. Deflation with several 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,

$$U_j \equiv [u_1, u_2, \ldots, u_j]$$

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 the following.

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*

$$A_j \equiv A - U_j \Sigma_j U_j^H,$$

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 (2.3) be the Schur factorization of A. We have

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

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

$$A_j U = U[R - E_j \Sigma E_j^H]$$

and the result follows.

∎

It is interesting to note that the preservation of the Schur vectors is analoguous to the preservation of the eigenvectors under Hotelling's deflation, in the symmetric case, see [35]. The above proposition suggests 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 y_{j+1} of A_j is computed by the appropriate algorithm $\mathcal{A}$ 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}. Clearly, u_{j+1} is a Schur vector associated with the eigenvalue λ_{j+1} and therefore at every stage of the process we have the desired decomposition

$$AU_j = U_j R_j, \tag{2.4}$$

where R_j is some j x j upper triangular matrix. The corresponding algorithm will be described in detail shortly.

With the above implementation, we may have to perform most of the computation in complex arithmetic. Fortunately, when the matrix A is real, this can be avoided. In that case the Schur form is traditionally replaced by the quasi-Schur form, in which one still seeks for the factorization (2.4) but simply requires that the matrix R_j, be quasi-triangular, i.e. one allows for 2 x 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, one important observation is 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, all we have to do is 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.

We can now sketch the Schur-Wielandt deflation procedure for computing the p eigenvalues of largest real parts.

Algorithm: Progressive Schur-Wielandt Deflation (PSWD)

(1) *Initialize:*

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

(2) *Compute next eigenvector (s) :*

Call algorithm $\mathcal{A}$ 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 $u_1, u_2, ...u_j$, to get u_{j+1}, (resp. u_{j+1}, u_{j+2}).

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

(4) *Test:*

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

A few additional details on the implementation of each step of the algorithm are now given. First, we point out that the above algorithm has as a parameter the algorithm $\mathcal{A}$, which delivers the eigenvalue(s) with largest real part(s) with its (their) associated eigenvector(s). We will discuss various choices of this algorithm in the next section. 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 $y_R \pm i y_I$. 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 brievety we have omitted to say in the algorithm that one need only to 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 Section 2.3, 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.

2.3. Error Analysis

In this section, we propose a few a posteriori error bounds in order to analyse the stability of the deflation technique. Typically, at each step $j = 1, 2, \ldots p$ 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

$$A_{j-1}y_j = \lambda_j y_j + \eta_j, \quad j = 1, \ldots p \tag{2.5}$$

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 $\lambda_j, j = 1, \ldots p$.

At step number j, the vector y_j is orthogonalized against $u_1, u_2, \ldots, u_{j-1}$ 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:

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

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

$$U_p b_j = y_j.$$

Replacing this relation in (2.5), we have

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

or

$$AU_p b_j = U_{j-1}\Sigma_{j-1}U_{j-1}^H U_p b_j + \lambda_j U_p b_j + \eta_j. \tag{2.6}$$

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

$$\Sigma_p \equiv \ Diag\ \{\sigma_1, \sigma_2, \ldots, \sigma_{p-1}, \sigma_p\}.$$

Then (2.6) becomes

$$AU_p b_j = U_p\,[\Sigma_p + (\lambda_j - \sigma_j)I]\,b_j + \eta_j, \quad j = 1, 2, \ldots, p.$$

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 $\Lambda_p \equiv Diag\{\lambda_1, \lambda_2, \ldots \lambda_p\}$. Then the above relation translates into the matrix relation:

$$AU_pB_p = U_p\,[\Sigma_p B_p + B_p(\Lambda_p - \Sigma_p)] + E_p, \tag{2.7}$$

which we rewrite in a final form as

$$AU_p = U_p\,[\Sigma_p + B_p(\Lambda_p - \Sigma_p)B_p^{-1}] + E_p B_p^{-1}. \tag{2.8}$$

For convenience, we define

$$Z_p \equiv E_p B_p^{-1} \tag{2.9}$$

and

$$C_p \equiv \Sigma_p + B_p(\Lambda_p - \Sigma_p)B_p^{-1}. \tag{2.10}$$

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

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

- Either one accepts the values $\lambda_i, i = 1, \ldots p$ 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 (2.10).
- 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 $\lambda_i, i = 1, \ldots p$ 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 $W \equiv [w_1, w_2, \ldots w_{N-p}]$ 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

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

in which $X_{12} = U_p^H AW, X_{22} = W^H AW$, 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:

$$A - [U_p, W] \begin{pmatrix} O & O \\ W^H Z_p & O \end{pmatrix} [U_p, W]^H = [U_p, W] \begin{pmatrix} R_p & X_{12} \\ O & X_{22} \end{pmatrix} [U_p, W]^H. \tag{2.11}$$

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 (2.11) establishes that the final result is equivalent to perturbing the initial matrix A by a matrix which is unitarily similar to the matrix

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

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

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

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:

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

from which we derive the upper bound

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

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

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

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

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

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

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

which is restated in the following proposition.

Proposition 2.3. *The Frobenius norms ρ_j of the matrices $Z_j, j = 1, \ldots p$ satisfy the recurrence relation*

$$\rho_j \leq (1 + cot\ \theta_j)\rho_{j-1} + \frac{||\eta_j||}{sin\theta_j}, \tag{2.12}$$

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 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 (2.5) 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

$$A_{j-1}u_j = \lambda_j u_j + \sum_{i=1}^{j-1} \gamma_{ij} u_i \quad + \eta_j. \tag{2.13}$$

Then an analysis similar to the one used to establish (2.8) 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 (2.13) 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 exists other methods that delivers directly a Schur vectors, is worth investigating.

3. Deflation techniques for three basic methods

In this section we review a few methods for computing eigenvalues and eigenvectors of large nonsymmetric matrices which can be used in the inner loops of algorithm PSWD of section 2.2. The methods are only briefly summarized as they have been fully described elsewhere in the litterature [1, 27, 26, 25].

3.1. Arnoldi's method with deflation

Arnoldi's method may not be considered as a powerful technique in itself but is a very useful tool when combined with other processes, such as the ones to be described in the next sections. Starting with some initial vector v_1 of Euclidean norm 1, the method generates the finite sequence of vectors by the recursion:

$$h_{j+1,j}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i, \quad j = 1, \ldots m, \tag{3.1}$$

where $h_{ij} = (Av_j, v_i), i = 1, \ldots j$ and $h_{j+1,j}$ is the 2-norm of the right hand-side of (3.1). The scalars h_{ij} are computed so that the sequence $v_1, v_2 \ldots v_m$ is an orthonormal sequence, in effect an orthonormal basis of the Krylov subspace $K_m = span\{v_1, Av_1, \ldots A^{m-1}v_1\}$.

Defining V_m as the N x m matrix whose i^{th} column is the vector v_i, for $i = 1, 2, \ldots m$ and H_m as the upper Hessenberg matrix whose entries are the coefficients h_{ij} computed during Arnoldi's method, a simple consequence of the relation (3.1) is that

$$V_m^H A V_m = H_m. \tag{3.2}$$

Therefore the eigenvalues of H_m constitute the Galerkin approximations of the eigenvalues of A on the Krylov subspace. Moreover, the corresponding approximate eigenvectors are given by

$$y_i^{(m)} = V_m z_i^{(m)}, \tag{3.3}$$

where $z_i^{(m)}$ is an eigenvector of the Hessenberg matrix H_m associated with the eigenvalue $\lambda_i^{(m)}$. This was the basis of Arnoldi's original method presented in [1]. For some details on theory and practical use of this process see [27].

A major limitation of Arnoldi's method is that its cost and storage requirements increases drastically as the number of steps m required for convergence increases. An immediate remedy for this is to use restarting: after the m steps are performed one restarts with an initial vector formed from a linear combination of the eigenvectors (3.3) associated with the desired eigenvalues. This was proposed as a simple alternative to the classical method in [27] and was further improved by the incorporation of polynomial-based acceleration techniques in [26] and [25].

However, the restarting method may encounter some difficulties especially in cases when the number of wanted eigenvalues is not small. One way in which this inefficiency manifests itself is that when restarting, the process is often unable to keep the accuracy gained in the previous steps for all eigenvalues, i.e., the accuracy may improve in some eigenvalues but deteriorates in some others. It is difficult when the number of eigenvalues is not small to make the method produce a similar accuracy for all the wanted eigenvalues. This is why deflation is so important. Very often it is possible to recover convergence by using a larger number of steps in the iterative Arnoldi method, but this is not always desirable as the storage requirement increases drastically.

Since Arnoldi's method is relatively successful in computing the eigenvalue of largest real part of a large nonsymmetric matrix, we can improve the reliability of the method by always computing one eigenvalue and its eigenvector at a time, i.e. by never attempting to extract more than one eigenpair at a time. This can be achieved by using the deflation algorithm PSWD, in which algorithm $\mathcal{A}$ is simply replaced by the restarted Arnoldi.

3.2. The Chebyshev - Arnoldi method with deflation

One way of avoiding the weaknesses of Arnoldi's method is to use a good initial vector, i.e. an initial vector for the total number of Arnoldi iteration is small. This can be achieved by preprocessing the initial vector by a polynomial type iteration before feeding it into the Arnoldi algorithm. The question of course, is how to select a good polynomial. The basic principle of polynomial acceleration techniques is to start by enclosing the set of unwanted eigenvalues in some domain and then find a polynomial which has a small modulus in that domain comparatively to its modulus on the wanted eigenvalues. In Chebyshev acceleration the enclosing domain is an ellipse and the basic idea is to minimize the maximum modulus of a polynomial p over that ellipse subjset to the constraint that $p(\lambda_1) = 1$, where λ_1 is the eigenvalue of largest real part, assumed to be real here for simplicity. This approach which leads naturally to Chebyshev polynomials, was considered by Manteuffel who uses it as the basis for an iterative method for solving nonsymmetric linear systems [15, 16].

In [26] we have described a hybrid method based on a combination of Chebyshev iteration and Arnoldi's method for computing eigenvalues and eigenvectors of nonsymmetric matrices. The algorithm described in [26] attempts to compute *simultaneously* the set of p eigenvalues of largest (or smallest) real parts. Schematically, the algorithm runs as follows. Initially, we perform m steps, of Arnoldi's method where $m>p$ is fixed, starting with a random initial vector. This computes a set of m approximate eigenvalues which are split in two parts: the set of wanted eigenvalues (i.e. the p approximate eigenvalues with largest real parts) and that of the unwanted ones (the remaining approximate eigenvalues). From the set of unwanted eigenvalues, one builds a polygonal convex hull that contains all the unwanted eigenvalues. Then the parameters of an ellipse containing that convex hull are computed. The parameters of the computed ellipse are optimal in a certain sense. Using these parameters, a certain number of steps of Chebyshev iteration are performed starting with a certain linear combination of the approximate Arnoldi eigenvectors associated with the wanted eigenvalues, as an initial vector. The effect of the Chebyshev iteration, is to damp the eigencomponents associated with the eigenvalues *inside* the ellipse contaning the convex hull of unwanted eigenvalues while highly amplifying those components associated with the wanted eigenvalues. As a result the final vector of this Chebyshev iteration is a perfect candidate for

Arnoldi's method. We can then reproduce the previous steps with this vector as v_1. This process is repeated until the whole set of desired eigenvalues has converged.

This combination constitutes an extremely powerful technique if only the eigenvalue of largest real part is to be computed. When several of them must be computed then the restarting procedure may lead to some difficulties. See [26] for details on this process and numerical experiments. Again reliability will be improved by simply computing one eigenpair at a time and using the Schur-Wielandt deflation.

3.3. The Least Squares - Arnoldi method with Deflation

The choice of ellipses as enclosing regions in Chebyshev acceleration may be overly restrictive and ineffective if the shape of the convex hull of the unwanted eigenvalues bears little resemblance with an ellipse. This was the motivation of the work [25] in which the acceleration polynomial is chosen so as to minimize an L_2-norm of the polynomial p on the boundary of the convex hull of the unwanted eigenvalues with respect to some suitable weight function ω. The only restriction with this technique is that the degree of the polynomial is limited because of cost and storage requirement. This, however, is overcome by compounding low degree polynomials. The stability of the computation is enhanced by employing a Chebyshev basis and by a careful implementation in which the degree of the polynomial is taken to be the largest one for which the Gram matrix has a tolerable conditioning. The method for computing the least squares polynomial is fully described in [25] but we present a summary of its main features below.

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

$$z_k = p_k(A) z_0 \tag{3.4}$$

where p_k is a degree k polynomial. If A is diagonalizable, then by writing the expansion of z_0 in the basis of eigenvectors u_i of A as

$$z_0 = \sum_{i=1}^{N} \xi_i u_i$$

we get $z_k = \sum_{i=1}^{N} \xi_i p_k(\lambda_i) u_i$ which we separate in two parts

$$z_k = \sum_{i=1}^{r} \xi_i p_k(\lambda_i) u_i + \sum_{i=r+1}^{N} \xi_i p_k(\lambda_i) u_i \tag{3.5}$$

The principle of the hybrid least squares-Arnoldi method is to use the vector z_k as an initial vector. From the analysis of the Arnoldi process, it is clear that we want the second part of the above expansion to be small compared with the first part. In fact it can be proved [26] that if the second part is zero then the Arnoldi process will stop at the r^{th} step with K_r becoming the invariant subspace associated with the eigenvalues $\lambda_1, \lambda_2, \ldots \lambda_r$. Therefore, we wish to choose among all polynomials p of degree $\leq k$ one for which $p(\lambda_i), i > r$ are small relative to $p(\lambda_i), i \leq r$.

Assume that by some adaptive process, a polygonal region H which encloses the remaining eigenvalues becomes available to us. We then arrive at a function approximation problem, which roughly formulated consists of finding a polynomial of degree k whose value inside some region is small while its values at r particular points (possibily complex) are large. For a more precise formulation we begin by normalizing the polynomial at the points $\lambda_1, \lambda_2, \ldots \lambda_r$. One such normalization is

$$\sum_{j=1}^{r} \mu_j p(\lambda_j) = 1 \tag{3.6}$$

in which the $\mu_j's, j = 1, \ldots r$ constitute r different weights. Since it is known that the maximum modulus of an analytic function over a region of the complex plane is reached on the boundary of the region, one solution to the above problem is to minimize an L_2-norm associated with some weight function ω, over all polynomials of degree k satisfying the constraint (3.6). We need to choose a weight function ω that leads to easy computations in practice.

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

$$\omega_j(\lambda) = \frac{2}{\pi}|d_j^2 - (\lambda - c_j)^2|^{-1/2} \tag{3.7}.$$

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

$$< p, q >_\omega = \int_{\partial H} p(\lambda)\overline{q(\lambda)}\omega(\lambda)|d\lambda| = \sum_{j=1}^{\mu} \int_{E_j} p(\lambda)\overline{q(\lambda)}\omega_j(\lambda)|d\lambda| \tag{3.8},$$

and the corresponding L_2-norm is denoted by $||.||_\omega$. Then we have the following result [25].

Theorem 3.1. *Let $\{\pi_i\}_{i=0,k}$ be the first $k+1$ orthonormal polynomials with respect to the L_2-inner-product (3.8). Then among all polynomials p of degree k satisfying the constraint (3.6), the one with smallest ω-norm is given by*

$$p_k(\lambda) = \frac{\sum_{i=0}^{k} \Phi_i \pi_i(\lambda)}{\sum_{i=0}^{k} |\Phi_i|^2} \tag{3.9}$$

where $\Phi_i = \sum_{j=1}^{r} \bar{\mu}_j \overline{\pi_i(\lambda_j)}$.

On the practical side the process of constructing the orthogonal polynomials can be difficult and unstable if not enough care is exerted in the computation. In [25] and [24], the orthogonal polynomials as well as their linear combination (3.9) are all expressed in terms of a Chebyshev basis associated with the ellipse of smallest area containing H. Then, the moment matrix M_k associated with this basis is constructed *without any numerical integration.* The solution (3.9) is then obtained by solving a linear system with this Gram matrix. The tedious details on the practical computation may be found in [25].

The resulting hybrid method for computing the r eigenvalues with largest real parts is outlined next.

The Hybrid Least-Squares Arnoldi Algorithm

(1) *Start:*

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

(2) *Projection step:*

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

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

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

Obtain the new least squares polynomial of degree k.

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

(3) *Polynomial iteration:*

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

Many practical details are omitted and are discussed at length in [25]. We only mention that the linear combination at the end of step 3, is usually taken as follows:

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

in which the vectors $\tilde{u}_i$ are the normalized approximate eigenvectors and ρ_i are their residual norms. The effect of this heuristic choice is twofold. First it avoids complex arithmetic when the matrix A is real, because then the vector z_0 is always real. Second, it avoids the damaging effects of unbalanced convergence by putting more emphasis on the eigenvectors that are slower to converge. In fact here lies a weaknesses similar to that of the restarted Arnoldi method mentioned in section 3.1. It is difficult to choose a linear combination that leads to balanced convergence because it is difficult to represent a whole subspace by a single vector. This translates into divergence in many cases especially when the number of wanted eigenvalues r is not small. There is always the possibility of increasing the space dimension m, at a high cost, to ensure convergence but this solution is not always satisfactory from the practical point of view.

Use of deflation constitutes a good remedy against this difficulty because it allows us to compute one eigenvalue at a time which is much easier than computing a few of them at once. Another solution is to improve the separation of the desired eigenvalues by replacing A by a polynomial in A. This approach, referred to as polynomial preconditining will be presented in the next section.

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

4. A Polynomial Preconditioned Arnoldi Method

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. If the shift σ is suitably chosen the shifted and inverted matrix $B = (A - \sigma I)^{-1}$, has a spectrum with much better separation properties than the original matrix A and therefore would require 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 B is so much less than that required with A that the cost of factorization is payed off. Essentially the same argument is used in the preconditioned conjugate gradient method when dealing with linear systems.

There are instances where shift and invert is essential and should not be avoided, as for example for the generalized eigenvalue problems $Ku = \lambda Mu$. The reasons are discussed at length in [18], [17], [23] and [31] 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.

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

$$B_k = p_k(A) \tag{4.1}$$

where p_k is a degree k polynomial. Ruhe [22] considers a more general method in which p_k is not restricted to be a polynomial but can be a rational function. When an Arnoldi type method is applied to B_k, we do not need to form B_k explicitly, since all we will ever need in order to multiply a vector x by the matrix B_k is k matrix-vector products with the original matrix A and some linear combinations.

For fast convergence, we would ideally like that the r wanted eigenvalues of largest real parts of A be transformed by p_k into r eigenvalues of B_k that are very large as compared with the remaining eigenvalues. Thus, we can proceed as in section 3.3, by attempting to minimize some norm of p_k in some region subject to the constraint (3.6). Once again we have freedom in choosing 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. Therefore, we are faced again with the function approximation problem described in Section 3.3.

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

eigenvalues of A are the eigenvalues of A_m and the approximate eigenvectors are given by $V_m y_i^{(m)}$ where $y_i^{(m)}$ is an eigenvector of A_m associated with the eigenvalue $\tilde{\lambda}_i$. A sketch of the algorithm is as follows.

The Preconditioned Arnoldi-Least Squares Eigenvalue Algorithm

(1) *Start:*

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

(2) *Initial Arnoldi Step:*

Using the initial vector v_1, perform m steps of the Arnoldi method with the matrix A.

(3) *Projection Step:*

Obtain the matrix $A_m = V_m^T A V_m$ and its m eigenvales $\{\tilde{\lambda}_1, \ldots \tilde{\lambda}_m\}$ and eigenvectors $\tilde{y}_i$.

Compute the approximate eigenvectors $\tilde{u}_i = V_m \tilde{y}_i$ for $i = 1, 2 \ldots r$ and their residual norms $\rho_i, i = 1, r$. If satisfied then Stop. Else

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

Obtain the new least squares polynomial p_k of degree k.

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

(4) *Arnoldi iteration:*

Perform m steps of Arnoldi's method with the matrix $B_k = p_k(A)$ starting with $v_1 = z_0/||z_0||$. Goto 3.

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. The linear combination of the approximate eigenvectors in step 3 is the same as that of the Hybrid Arnoldi/Least squares method of Section 3.3.

The difference between this method and that of section 3.3 is that here the polynomial iteration is an inner iteration and Arnoldi is the outer loop, while in the hybrid method, the two processes are serially following each other. Both methods can be viewed as means of accelerating the Arnoldi method.

It is clear that the Schur-Wielandt deflation technique can also be applied to the polynomial preconditioned Arnoldi method and this is recommended. However, in our numerical experiments, we will only compare the approach of deflation in conjunction with polynomial acceleration in the sense of the previous sections, versus that of polynomial preconditioning without any deflation.

5. Numerical experiments

All numerical tests have been performed on a Vax-785 using double precision, i.e., the unit roundoff is $2^{-56} \approx 1.3877 \text{ x } 10^{-17}$.

5.1. The test example

Our test example, taken from [21], 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: [21]:

$$\frac{\partial x}{\partial \tau} = \frac{D_x}{L^2}\frac{\partial^2 x}{\partial z^2} + f(x,y), \tag{5.1}$$

$$\frac{\partial y}{\partial \tau} = \frac{D_y}{L^2}\frac{\partial^2 y}{\partial z^2} + g(x,y), \tag{5.2}$$

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 (5.1) - (5.2) evaluated at the steady state solution. In many problems one is primiraly interested in the existence of limit cycles, or equivalently the existence of periodic solutions to (5.1), (5.2). This translates into the problem of determining whether the Jacobian of (5.1), (5.2) evaluated at the steady state solution admits a pair of purely imaginary eigenvalues.

We consider in particular the so-called Brusselator wave model [21] in which

$$f(x,y) = A - (B+1)x + x^2 y \tag{5.3}$$

$$g(x,y) = Bx - x^2 y. \tag{5.4}$$

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 (5.1), (5.2) 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 [21] 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 complex 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 $\begin{pmatrix} x \\ y \end{pmatrix}$ 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 $Tridiag\{1,-2,1\}$.

For reference we name ARNIT the iterative Arnoldi method of section 3.1, CHBARN the Arnoldi-Chebyshev method with deflation of Section 3.2, LSARN the least squares polynomial method combined with Arnoldi of Section 3.3 and ARNLS the least squares preconditioned Arnoldi method of section 4.

5.2. Computing one pair of eigenvalues

In this first test we compare the four methods described earlier to compute the pair of eigenvalues having largest real parts of the $2n$ x $2n$ matrix A. We used a discretization of $n = 100$ subintervals, i.e., the size the resulting matrix is 200. We then ran the four methods with a size m of Arnoldi dimension equal to 20, in all cases, and in either CHBARN or LSARN the maximum degree polynomial was 100. For ARNLS the degree of polynomials was chosen 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. We have set the parameter indicating the number of wanted eigenvalues to $NEV = 1$. Note that here the eigenvalue of largest real part is complex, in fact almost exactly purely imaginary, so a reasonable code should deliver a pair of complex conjugate eigenvalues in this situation.

The residual norms provided by the first three methods which deal with A are not comparable with those provided by the method ARNLS which deals with B_k, a polynomial in A. We therefore opted to compare the computed eigenvalues during the various runs with the exact ones which are known. The exact eigenvalues, determined with maximum accuracy (double precision), i.e. approximately 16 digits are

$$\lambda_{1,2} = 1.8199876787305946 \text{ x } 10^{-5} \pm i\ 2.139497522076329$$

As is observed the real part is close to zero, which verifies the theory, within the discretization errors. We have plotted the relative errors

$$\left| \frac{\lambda_1^{(m)} - \lambda_1}{\lambda_1} \right| \tag{5.5}$$

for each of the 4 methods.

For ARNLS, the preconditioned Arnoldi method, the approximate eigenvalue $\lambda_1^{(m)}$ was determined as the Rayleigh quotient $(A\phi,\phi)/(\phi,\phi)$ obtained from the approximate eigenvector ϕ of the matrix B_k, which is easily computed within the Arnoldi process which is applied to B_k. This is done every five Arnoldi loops. For the other methods, the error is plotted after each adaptive

Arnoldi step. All methods are started with the same initial vector which is a random vector. The results are plotted in figure 1.

ARNIT did not show any sign of convergence after a total of 1000 matrix by vector multiplications. LSARN and CHBARN perform similarly while ARNLS differs significantly in that it starts more slowly than both LSARN and CHBARN but as soom as a good convex hull of the eigenvalues is found, it outpaces all the other methods. Note that for simplicity we have not applied any complicated heuristic such as varying the Arnoldi dimension m from lower to high values in order to build the convex hull H more gradually. The ICYCLE paramater in the figure shows the degree of polynomials used in both CHBARN and LSARN. In this test this was set to 100. However, for LSARN, the polynomial of degree 100 is obtained by compounding (5 times) a least squares polynomial of degree 20, as is indicated in the figure.

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 that 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 Fobenius norm $f_m =$ Trace $(H_m^H H_m)$, and take as an estimate of the error of the computed pair eigenvalue/eigenvector the number

$$\frac{\rho}{\sqrt{\frac{1}{m} f_m}}, \tag{5.6}$$

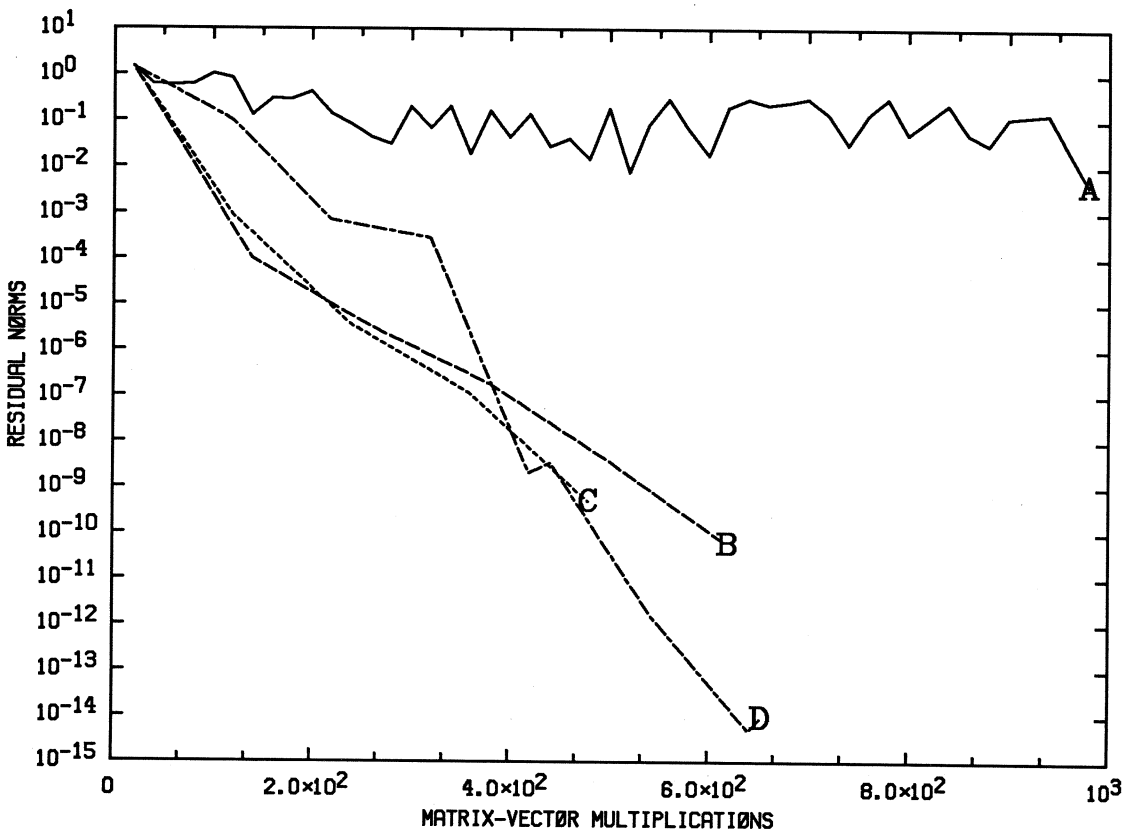
where ρ is the computed residual norm provided by the method. Note that the denominator represents the square root of the average of the squares of the singular values of H_m. In ARNLS the same scaling is used except that for the projection step (Step 4 of Algorithm ARNLS), H_m is replaced by the matrix A_m. Recall [27] 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.

With this stopping criterion, and $\epsilon = 10^{-7}$ the method stopped in the order indicated in Figure 1, i.e. LSARN stopped first with a total of 480 matrix multiplications, then CHABRN (total of 620 matrix vector multiplications) closely followed by ARNLS (total of 654 matrix vector multiplications) and then ARNIT (no convergence). Thus, it is clear from this example that the residual norms do not reflect the actual errors in the eigenvalues. The plot shows for example that the error estimate (5.6) is an overestimate of the actual error in all cases (since the method continued to run well after this estimate went below the 10^{-7} mark in the plot). The intriguing fact is that it more pessimistic for the matrix B_k than it is for the matrix A. More precisely, at the end of the run of ARNLS the estimate (5.6) was of the order of 1.91 x 10^{-8} while, as is indicated by the plot, the error on the eigenvalue is 9.63 x 10^{-15}. As a comparison, CHBARN and LSARN showed a smaller discrepancy: the estimate is 9.1710^{-8} versus the actual value 5.98 x 10^{-11} for CHBARN and 9.29 x 10^{-8} versus 5.13 x 10^{-10} for LSARN. This phenomenon shows that the preconditioning technique does actually improve the conditioning of the eigenpair: the actual error is almost of the order of the square of the residual norm based error estimate, just like for symmetric matrices.

As is shown in the next experiment the performances of the above methods depend critically on the values of the parameters m (dimension of Krylov subspace in Arnoldi's method) and k the degree of the accelerating polynomial. In the next plot we shows the same experiment as in the previous one except that the Arnoldi dimension m, is set to 30, instead of 20.

5.3. Computing several eigenvalues

In this test we compute the 6 rightmost eigenvalues of the same 200 x 200 matrix A as in the previous test. These 6 rightmost eigenvalues form three complex conjugate pairs. Using the same



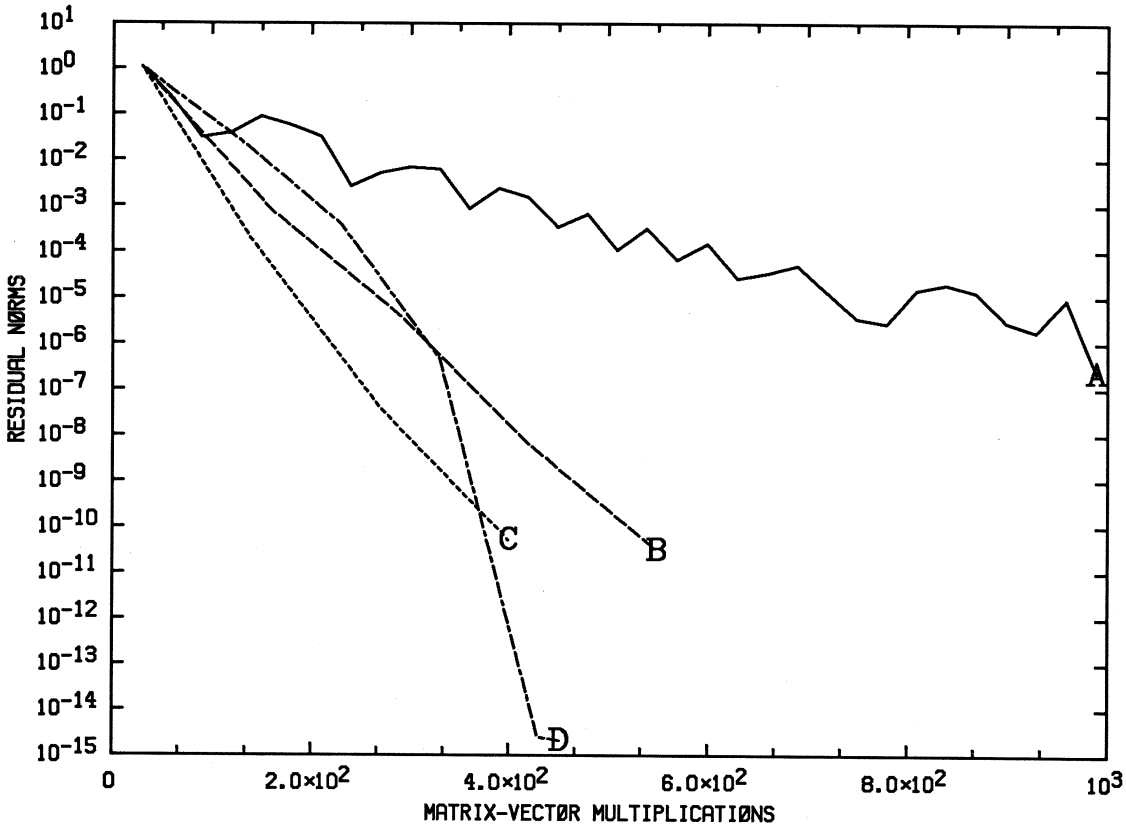
A ARNIT (ITERATIVE ARNØLDI), IARN = 20

B CHBARN (CHEBYSHEV-ARNØLDI), IARN = 20, ICYCLE = 100

C LSARN (LEAST SQUARES/ARNØLDI), IARN = 20, ICYCLE = 5X20

D ARNLS (LS/PRECØNDITIØNED ARNØLDI), IARN = 20, NDEG = 20

Figure 1: Computing one pair with 4 different methods for $N = 200$.



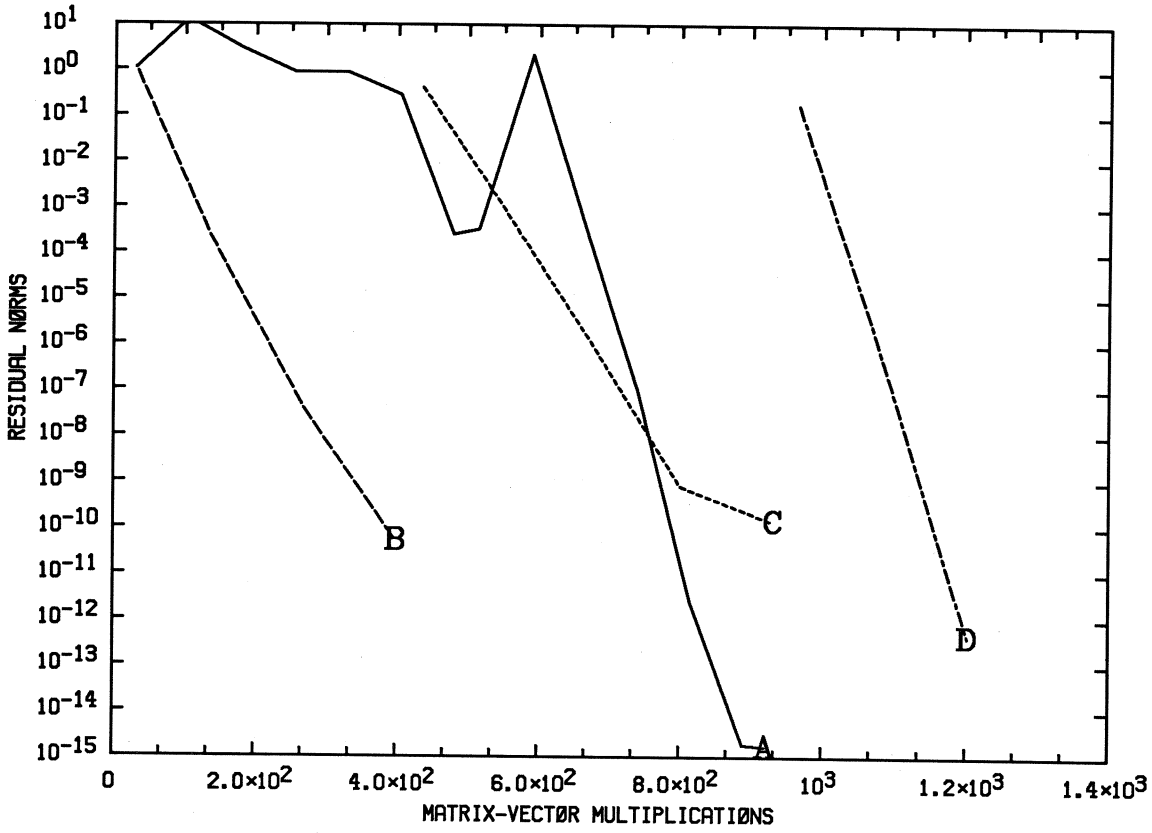
A ARNIT (ITERATIVE ARNØLDI), IARN = 30

B CHBARN (CHEBYSHEV-ARNØLDI), IARN = 30, ICYCLE = 100

C LSARN (LEAST SQUARES/ARNØLDI), IARN = 30, ICYCLE = 5X20

D ARNLS (LS/PRECØNDITIØNED ARNØLDI), IARN = 30, NDEG = 20

Figure 2: Computing one pair of eigenvalues with 4 different methods for $N = 200$.



A ARNLS (IARN = 30, NDEG=15)

B LSARN (IARN = 30, ICYCLE=5X20), FIRST PAIR

C SECOND PAIR

D THIRD PAIR

Figure 3: Computing three pairs of eigenvalues with two methods for $N = 200$.

j	$\|\|Z_j\|\|$	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.

stopping criterion as before it took 922 matrix by vector multiplications for the method ARNLS to converge with $m = 30$ and $k = 15$. As a comparison it took 1204 such multiplications for the method LSARN used with deflation to deliver all the three pairs of eigenvalues. (400 for the first pair 532 for the second pair and 272 for the last pair). The Chebyshev/Arnoldi method performed similarly, taking a total of 1264 matrix-vector multiplications for delivering the three pairs. In the graph of Figure 3, we have plotted the convergence history of the two methods ARNLS and LSARN. In LSARN the degree of polynomials is 100, i.e., we compound 5 times a polynomial of degree 20. Three curves corresponding to the convergence history of each of the three pairs of eigenvalues are drawn. As before we have plotted the relative error (5.5) of the computed eigenvalue, versus the accumulated number of matrix by vector multiplications during the run.

For the method ARNLS the eigenvalues are computed simultaneously and we have therefore graphed the average of the relative errors, over the 3 pairs. Here we have taken $m = 30$ and the degree k is set to 15. Since there is little difference between the Chebyshev method CHBARN and the least-squares method LSARN, we have omitted to plot the results obtained with CHBARN.

5.4. The Frobenius norm error bound

In this test we verify the error bound (2.12) of section 2.3 and in particular show how close the estimate can be. The test matrix is the same as in the preceding tests 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 only one method as Algorithm $\mathcal{A}$, namely LSARN with $m = 10$, and polynomial of degree $100 = 5 \times 20$. 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 (2.5) 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 (2.12). 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.

6. Summary and Conlcusion

We have presented two classes of methods for computing a few eigenvalues and the corresponding eigenvectors or Schur vectors of large nonsymmetric matrices. The first comprises a deflation method combined with any type of polynomial iteration. The second can be viewed as a preconditioned Arnoldi method, whereby one uses Arnoldi's method to iterate with a polynomial in A instead of A itself. 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 diferential equations, or in the analysis of various bifurcation phenomena. In those problems, a (few) rightmost eigenvalues of some Jacobian matrix must be computed in continuation type techniques. It is clear that the information gathered from previous continuation steps can be used if the marching parameter varies slowly: in this fashion a good initial vector for the next run is available as well as a good convex hull of the unwanted eigenvalues and one can expect a relatively moderate extra work at each new continuation step.

The deflation technique can also be of great help when dealing with the generalized eigenvalue problem. There, if one uses an Arnoldi (or nonsymmetric Lanczos) method, big savings can be made by using deflation because it allows to go farther in the spectrum without having to perform a new factorization of $K - \sigma M$ too soon. In essence the selective orthogonalization technique developed by Parlett and Scott [17, 30] realizes a similar deflation technique in the symmetric case in a more economical way. Our analysis of section 2.3 and our experiments indicates that the Schur-Wielandt deflation is safe to use. The a-posteriori upper bound of of Proposition 2.3, can be used in practice to determine how accurate a computed basis of an invariant subspace is.

References

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

[2] F.L. Bauer, *Das Verfahren der Treppeniteration und Verwandte Verfahren zur Losung Algebraischer Eigenwertprobleme,* ZAMP, 8 (1957), pp. 214–235.

[3] P.N. Brown, A.C. Hindmarsh, *Matrix-free methods for stiff systems of ODE'S,* Technical Report UCRL-90770, Laurence Livermore Nat. Lab., 1984.

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

[5] T.F. Chan and K. R. Jackson, *The use of iterative linear equation solvers in codes for large systems of stiff IVPs for ODEs,* Technical Report 170/84, Univ. of Toronto, 1984.

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

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

[8] L. E. Eriksson and A. Rizzi, Analysis by computer of the convergence of discrete approximations to the Euler equations, *Proceedings of the 1983 AIAA conference, Denver 1983,* AIAA paper number 83-1951, Denver, 1983, pp. 407–442.

[9] W.C. Gear and Y. Saad, *Iterative solution of linear equations in ODE codes,* SIAM J. Sci. Stat. Comp., 4 (1983), pp. 583–601.

[10] H. Hlavacek and H. Hofmann, *Modeling of Chemical reactors XVI,* Chemical Eng. Sci., 25 (1970), pp. 1517–1526.

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

[12] A. Jepson, *Numerical Hopf Bifurcation,* Ph.D. Thesis, Cal. Inst. Tech., 1982.

[13] D.D. Joseph, D.H. Sattinger, *Bifurcating time periodic solutions and their stability,* Arch. Rat. Mech. An, 45 (1972), pp. 79–109.

[14] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operator,* J. Res. Nat. Bur. of Standards, 45 (1950), pp. 255–282.

[15] T.A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems,* Numer. Mat., 28 (1977), pp. 307–327.

[16] ————, *Adaptive procedure for estimation of parameter for the nonsymmetric Tchebychev iteration,* Numer. Mat., 28 (1978), pp. 187–208.

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

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

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

[20] B.N. Parlett, D. R. Taylor, Z.S. Liu, The look ahead Lanczos algorithm for large nonsymmetric eigenproblems, J.L. Lions and R. Glowinski ed., *Proceedings of the 6-th International Conference on Computing Methods in Engineering and Applied Sciences, Versailles, France, Dec. 12-16 1984,* INRIA, North-Holland, 1985.

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

[22] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computations,* Linear Algebra and its Applications, 58 (1984), pp. 391–405.

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

[24] Y. Saad, *Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems,* SIAM J. Stat. Sci. Comput., (-). submitted.

[25] ————, *Least squares polynomials in the complex plane with applications to solving sparse nonsymmetric matrix problems,* Technical Report 276, Yale University, Computer Science Dept., 1983.

[26] ————, *Chebyshev accelaration techniques for solving nonsymmetric eigenvalue problems,* Mathematics of Computation, 42 (1984), pp. 567–588.

[27] ————, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices,* Lin. Alg. Appl., 34 (1980), pp. 269–295.

[28] ————, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems,* SIAM j. Numer. Anal., 19 (1982), pp. 470–484.

[29] G. Sander, C. Bon, M. Geradin, *Finite element analysis of supersonic panel flutter.,* Int. J. Num. Meth. Engng., 7 (1973), pp. 379–394.

[30] D.S. Scott, *Analysis of the symmetric Lanczos process,* Ph.D. Thesis, University of California at Berkeley, 1978.

[31] ————, *The advantages of inverted operators in Rayleigh-Ritz approximations,* SIAM J. on Sci. and Stat. Comp., 3 (1982), pp. 68–75.

[32] G.W. Stewart, *Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices,* Numer. Mat., 25 (1976), pp. 123–136.

[33] ————, *SRRIT - a FORTRAN subroutine to calculate the dominant invariant subspaces of a real matrix,* Technical Report TR-514, University of Maryland, 1978.

[34] D. Taylor, *Analysis of the look-ahead Lanczos algorithm,* Technical Report, Univ. Calif. Berkeley, 1983. PhD thesis.

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