JOURNAL OF COMPUTATIONAL PHYSICS 100, 128–142 (1992)

An Arnoldi-Based Iterative Scheme for Nonsymmetric Matrix Pencils Arising in Finite Element Stability Problems

RAMESH NATARAJAN

IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598

Received March 23, 1990; revised August 27, 1990

A method for computing the desired eigenvalues and corresponding eigenvectors of a large-scale, nonsymmetric, complex generalized eigenvalue problem is described. This scheme is primarily intended for the normal mode analysis and the stability characterization of the stationary states of parameterized time-dependent partial differential equations, in particular, when a finite element method is used for the numerical discretization. The algorithm, which is based on the previous work of Saad, may be succintly described as a multiple shift-and-invert, restarted Arnoldi procedure which uses reorthogonalization and automatic shift selection to provide stability and convergence, while minimizing the overall computational effort. The application and efficiency of the method is illustrated using two representative test problems.

1. INTRODUCTION

In this paper, we consider some numerical algorithms that are useful in the analysis of the dynamical behavior of physical systems that are described by parameterized nonlinear partial differential equations. The steady states of such systems are governed by certain linearized perturbation equations, whose discretization by a finite element method leads to a large matrix eigenvalue problem. As discussed in detail below, the overall dynamics is then dominated by the few modal perturbations whose eigenvalues have the largest real parts. In practice, for realistic applications, a great deal of program development and large computational resources are required in order to obtain these stability determining eigenvalues. Therefore, it is important to develop algorithms that are general and robust, for use without modification on a wide range of problems. Equally, it is important for algorithms to be well-matched to the capabilities of current high-performance vector and parallel computers, which will inevitably be required for tackling such applications.

The routines in EISPACK and other general-purpose software libraries have some well-known limitations that make them inappropriate for these intended problems. First, these routines require random access to the data, and therefore work best with small matrices that fit entirely in core memory. Second, they provide no particular computational saving if the matrices are sparse and if only a small fraction of the eigenspectrum is desired. Third, the entire computation has to be repeated from scratch for each separate problem instance, and no use is made of any information that may have been generated from a previous computation at a nearby value of the problem parameters. Fourth and finally, these routines are incapable of realizing the potential performance of current high-performance computers without non-trivial modifications, making it worthwhile to look at alternative methods that, at least conceptually, might vectorize or parallelize more readily.

In this context, the use of spectral methods for the discretization of continuous eigenvalue problems would appear to be quite attractive [5]. Here the choice of global basis functions for the discretization leads to small dense matrices (rather than the large, sparse matrices for finite element discretization of equivalent accuracy), so that the use of conventional eigenvalue software can be quite appropriate. However, these matrices frequently have a block non-zero structure that is not well exploited by algorithms intended for fully dense matrices. We note that the advantages of a finite element discretization lie in the much greater flexibility in handling boundary conditions and irregular problem geometries and, generally speaking, in the reduced programming effort required for each new application.

Recently, some other investigators have also considered the use of matrix eigenvalue algorithms for finite element applications of related interest. Jackson [7] studied the exterior two-dimensional flow of an incompressible fluid past bodies of various shapes, using a subspace iteration method to compute the eigenvalues in order to delineate the regimes of flow instability. Christodolou and Scriven [1] considered the stability behavior of free surface flows in viscous thin films by tracking the potentially unstable eigenvalues, using a scheme based on Arnoldi's method as

0021-9991/92 $5.00

described by Saad [12]. A third distinct approach is the unsymmetric Lanczos algorithm, which has been used by Cullum, Kerner, and Willoughby [2] in a plasma physics application. These three matrix eigenvalue algorithms have the property that they provide "iterative" approximations to several eigenpairs simultaneously, rather than one at a time as in the usual power or inverse iteration methods [14].

Of the three competing algorithmic strategies described above, we have chosen the Arnoldi-based scheme as representing the best compromise in terms of numerical accuracy, algorithmic efficiency, and programming simplicity between the other two methods. In our implementation, we have enhanced the basic procedure in various ways over the previous work, in order to improve its performance and utility for our intended applications. These modifications include the use of automatic shift selection heuristics, for avoiding stagnation, and for guiding the algorithm towards computing the eigenvalues of greatest interest, as well the use of a simple inverse iteration procedure for extracting the eigenvectors of specified computed eigenvalues at low cost. In addition, our implementation is based on the use of complex arithmetic, which simplifies the analytical effort and programming for a large class of relevant applications (although it has some other ramifications too, as discussed below).

The primary motivation for this research derives from the fact that classical methods in stability analysis are limited to the consideration of "model" problems with various simplifications in the boundary conditions and problem geometries [3]. For more realistic problems, difficulties arise when the perturbation equations are complicated and, even more notably, when the base stationary state itself can only be computed numerically. The two test examples considered in the paper, though well-studied, exemplify each of these situations. In the case of the Orr–Sommerfeld equation, the base stationary state is very simple, but the perturbation equations, after some reduction, form a fourth-order eigenvalue differential equation with variable coefficients. In the case of the chemical reactor problem, the computation of the base state requires the solution of a complicated nonlinear problem, and much care is required in selecting an appropriate discretization and numerical scheme for computing this itself, even setting aside the complexity of analyzing the perturbation equations. These two test problems are quite representative of the more challenging open problems that one might want to solve, and good previous results are available for them that can be used to validate the correctness of our present computations. We have made no attempt in this work to try and obtain new results for these test applications, although to our knowledge, some of the computational results in Section 4 are quite new.

To fix ideas regarding these applications, consider a continuous physical system whose description is given by an operator evolution equation of the form

$$\frac{\partial U}{\partial t} = \mathscr{F}(U, \mu, \hat{\mu}), \qquad \text{on} \quad \Omega, \tag{1.1}$$

where U belongs to a suitably normed space of functions on the domain Ω, with time-independent boundary conditions on $\partial\Omega$. The nonlinear operator $\mathscr{F}$ is assumed to depend only on U and its spatial derivatives. The scalar μ is a distinguished problem parameter (such as the Reynolds number in the Navier–Stokes equations, or the Rayleigh number in the Boussinesq equations) by varying which the nature of the solutions U of (1.1) is changed. The remaining fixed parameters in the problem are denoted by $\hat{\mu}$, and these will be omitted below in order to simplify the notation.

The finite element discretization of (1.1) yields a discrete evolution equation of the form

$$M\frac{du}{dt} = f(u, \mu), \tag{1.2}$$

where u is a vector of nodal values approximating the function U, f is a nonlinear vector function, and M is the finite element "mass" matrix. For fixed μ, the stationary solution $\bar{u}$ of (1.2) will satisfy the equation

$$f(\bar{u}, \mu) = 0. \tag{1.3}$$

One approach to solving (1.3) is Newton's method. Starting with an initial guess $\bar{u}^0$, we successively compute

$$J(\bar{u}^n, \mu)\,\delta\bar{u}^n = -f(\bar{u}^n, \mu), \tag{1.4}$$

$$\bar{u}^{n+1} = \bar{u}^n + \delta\bar{u}^n, \tag{1.5}$$

where J denotes the Jacobian matrix, until the desired convergence is obtained. This iteration will diverge unless a good initial guess is available, but this difficulty can be overcome by a continuation method using the parameter μ; i.e., after the Newton iteration in (1.4) and (1.5) has converged, we solve

$$J(\bar{u}, \mu)\frac{\partial\bar{u}}{\partial\mu} = -\frac{\partial f}{\partial\mu}, \tag{1.6}$$

and a good initial guess for Newton's method at a nearby parameter value $\mu + \delta\mu$ is then given by

$$\bar{u}^0(\mu + \delta\mu) = \bar{u}(\mu) + \delta\mu\frac{\partial\bar{u}}{\partial\mu}. \tag{1.7}$$

The stability of the stationary state $\bar{u}(\mu)$ can be analysed

by taking small disturbances of the form $we^{\sigma t}$ and linearizing in (1.2) to obtain

$$Jw = \sigma Mw. \tag{1.8}$$

A necessary condition for the stability of the stationary state is that all the eigenvalues of (1.8) have negative real parts, so that the small modal perturbations will decay (sufficient conditions for stability require the consideration of finite-amplitude perturbations and cannot be obtained by the present approach). Parenthetically, we note that the perturbation vector w must satisfy homogenous boundary conditions, since the non-homogeneous terms in the boundary conditions of the original evolution equation are exactly satisfied by the stationary solution $\bar{u}$ itself.

In stability problems, the primary interest is often in determining the value of the parameter μ at which the least stable or "leading" eigenvalue has its real part just becoming zero, indicating that the corresponding eigenmode is on the verge of becoming unstable. If the imaginary part of the leading eigenvalue is simultaneously zero, then we have a regular bifurcation at this parameter value, and a new branch of stationary solutions will emerge from the current solution branch. The appearance of a regular bifurcation can in fact be deduced without an explicit eigenvalue analysis, by monitoring the determinant of the Jacobian matrix, which must change sign whenever an eigenvalue passes through zero. In particular, if a direct elimination method is used in (1.4) to compute the stationary solution $\bar{u}$, then the determinant can be easily evaluated as the running product of the pivots during elimination. However, if an iterative method is used then it is no longer straightforward to monitor the sign of the determinant of this Jacobian matrix, and an explicit eigenvalue analysis may then be necessary to detect the occurrence of a regular bifurcation. The other way in which a stationary state can lose stability is through a Hopf bifurcation, with a leading complex conjugate pair of eigenvalues just crossing the imaginary axis as μ is varied, giving rise to time-periodic solutions bifurcation from the stationary solution branch. This situation, again, can only be detected by an explicit eigenvalue analysis.

2. BACKGROUND

An excellent overview of several aspects of the finite element matrix eigenvalue problem has been given by Strang and Fix [13], particularly for the case when the matrices J and M are real-symmetric with M positive-definite. There, however, the eigenvalues are all real, and the Sturm sequence property provides a powerful tool for identifying and obtaining all the eigenvalues in a given real interval. In addition, the eigenvectors are all mutually orthogonal, and any set of such eigenvectors will form a linearly independent basis for the invariant subspace spanned by them. These two properties are most valuable for computational algorithms, but they do not carry over to the present class of problems. However, there are many other important issues that can guide the selection, implementation, and efficiency of the solution algorithms for our applications, and we list the most important among them below:

- The matrices J and M can be real, but the complex case is of great interest in stability applications. In particular, this latter possibility arises whenever "travelling wave" perturbations are allowed. These perturbations will break the spatial symmetry of the base stationary solution, and the Jacobian matrix for the eigenvalue problem (1.8) is not the same as that used in (1.4) to solve for the stationary solution by Newton's method. These complex-valued travelling wave disturbances can be phase shifted by the spatial derivative operators in the perturbation equations, leading to the appearance of complex coefficients in the eigenvalue problem (see, for example, the Orr–Sommerfeld equation in Section 5). Although in practice, it is always possible to formulate an equivalent real eigenvalue problem, this requires additional analysis and effort, so that it is most helpful to have programs that can handle the complex-valued case.

- The matrix M will in general be singular and non-invertible, with several rows having identically zero entries. This situation can arise from two contexts. First, this may be due to the absence of time derivative terms in some equations of the continuous problem formulation (a good example is the incompressibility condition on the fluid velocities in the Navier–Stokes equations). Second, and more commonly, this is due to the requirement that the perturbation vectors satisfy certain homogenous essential boundary conditions. These are most easily enforced by first generating the matrices J and M in the usual way, and then subsequently modifying them by setting all the entries in th rows of J and M corresponding to these boundary conditions to zero, except for the diagonal entry in J which is set to unity. This modification will lead to the required boundary conditions being automatically enforced in the solution procedure. In summary, therefore, the matrix M will have its rank reduced by the total number of discrete constraint equations and essential boundary conditions.

- The matrices J and M will be large and sparse, and by appropriately ordering the nodal unknowns in the finite element mesh, it is possible to obtain matrices of small bandwidth. These structural characteristics must be exploited for efficiently storing these matrices and for performing various matrix operations using them.

- If J and M are complex (which as mentioned earlier, is a case of considerable importance in applications) then the use of complex arithmetic in the implementation of the algo-

rithm cannot be avoided. However, the case when the matrices are real can also arise (as for example, in the chemical reactor example of Section 5), for which the eigenvalues and eigenvectors appear as complex conjugate pairs. Then with some additional programming it is possible to develop procedures for the basic iterative eigenvalue algorithms using only real arithmetic [14]. This will reduce the storage and computational costs in this part of the algorithm, but for reasons described below, will increase the computational cost in the shift-and-invert factorization. Parlett and Saad [10] have developed "hybrid" methods for real matrices that combine the performance benefits of both worlds. We have chosen to use complex arithmetic based procedures throughout, even for real-valued matrices, primarily because of the programming complexity of the real arithmetic-based algorithms, but we briefly note a technical issue that is relevant to this choice as well. For example, Cullum, Kerner, and Willoughby [2] have noted that the use of complex arithmetic in the Lanczos procedure lessens the possibility of the appearance of small divisors that lead to the numerical breakdown of the algorithm. Similarly, a source of numerical error in the Arnoldi iteration is the use of Gram–Schmidt orthogonalization for computing the Krylov subspace basis vectors (the more accurate modified Gram–Schmidt procedure cannot be used because of the incremental way in which this subspace is generated). Here, likewise, the use of complex arithmetic will provide some numerical robustness that can mitigate some of the increased storage and arithmetic costs.

• The finite element discretization of continuous eigenvalue problems often leads to the appearance of a large number of "spurious" eigenmodes in the discrete problem with corresponding eigenvalues having very large modulus. These spurious eigenmodes have no counterpart in the original continuous problem and are therefore extremely sensitive and ill-conditioned with respect to the discretization. This is a most important consideration, because the "factorization-free" versions of the iterative algorithms mentioned earlier, although computationally inexpensive, are only efficient in computing the extreme and well-separated eigenvalues in the complex plane. In our applications these are invariably the spurious eigenvalues. Even otherwise, the eigenvalues of greatest interest in applications are frequently located in the interior region of the spectrum. In stability problems, for example, these are the eigenvalues that either grow most rapidly or decay most slowly. While these eigenvalues do lie on the periphery of the spectrum in some sense, they are not necessarily the ones whose convergence is most favored in the factorization-free versions of any of these algorithms.

• In a practical application, the required eigenvalue computations will be performed in an "inner" loop, while in an "outer" loop the problem parameters are varied in order to map out the solution space. Therefore, an extrapolation procedure can be used to estimate the region of the complex plane that contains the desired eigenvalues and to enhance the convergence of the numerical algorithm for computing these eigenvalues.

We now review the basic ideas behind the algorithm keeping in mind the six important issues that we have outlined above. First, a shift-and-invert transformation is used, so that (1.8) is rewritten in the form

$$Kw = (J - \lambda M)^{-1} Mw = \hat{\sigma} w, \tag{2.1}$$

where the shift λ is a fixed complex number. This transformation has three very useful properties. First, the original generalized problem is transformed to a standard eigenvalue problem which has the same eigenvectors, but whose eigenvalues $\hat{\sigma}$ are given by

$$\hat{\sigma} = \frac{1}{\sigma - \lambda}. \tag{2.2}$$

Second, by suitably choosing the shift λ in the region of the complex plane which contains the eigenvalues of greatest interest, these are then mapped outwards in (2.2), so that the corresponding eigenvalues of K will have the largest modulus and the greatest separation and will therefore be the fastest to converge when the usual iterative algorithms are used. Third, by the same token, the "spurious" or infinite eigenvalues of (1.8) are mapped to the origin in the spectrum of (2.1), making them relatively benign in the transformed problem.

Since the solution algorithm for (2.1) described below requires only the product of the matrix K with a given vector, the shift-and-invert transformation can be applied by computing the LU factors of $J - \lambda M$. For complex J and M, using partial pivoting, the cost of computing this factorization is at most $N_t N_b^2$ in complex arithmetic, where N_t and N_b are the matrix order and bandwidth, respectively. However, if real arithmetic is used, then a matrix of order $2N_t$ and bandwidth $2N_b$ is obtained, and the computational cost for the factorization is increased by a factor of two, assuming that each complex arithmetic operation is equivalent to four real arithmetic operations (the actual ratio that we measured on a SUN 4/280 workstation was around 4.5). For large scale problems, the overall computational requirement will be dominated by the cost of this matrix factorization, so that the use of a complex arithmetic-based procedure for this is strongly recommended. We note that this is the case even when J and M are real matrices, since the shift λ is complex in general, so that the LU factors of K will be also be complex.

The action of K on a vector that is required in each iteration of the algorithm can be computed with one matrix-

vector product using M whose cost is $2N_tN_b$, and a forward and back solve using the LU factors whose cost is at most N_tN_b and $2N_tN_b$, respectively. The overall cost of this step is therefore within a small constant factor of the equivalent step without a shift-and-invert transformation. However, since the initial LU factorization can be expensive, and must be recomputed for each new choice of the shift value, it is best to try and compute as many eigenvalues as possible with a given shift, even if this requires a few additional iterations of the basic iterative eigenvalue algorithm.

The method that is used to compute some of the eigenvalues of the matrix K is a variant of the restarted Arnoldi method as described by Saad [12], in which the iteration vectors are maintained orthogonal to the invariant subspace of the converged eigenvectors. This reorthogonalization allows the Arnoldi algorithm to compute subdominant eigenvalues by arresting the tendency of the algorithm to repeatedly converge towards the dominant eigenvectors that have already been computed. In addition, in this way, the necessity for repeated factorizations of $J-\lambda M$ is reduced, since frequent new shifts are not required in order to uncover these subdominant eigenvalues. In effect, the cost of reorthogonalization is amortized by a more favorable convergence for the subdominant eigenvalues and by the fewer shifts that will be required to map out the spectrum.

Let us consider the case when an invariant subspace of order p for K has already been computed, so that

$$KU_p = U_pR_p, \tag{2.3}$$

where the columns of the unitary matrix U_p form a basis for this subspace and R_p is a complex upper triangular matrix, which represents the projection of K in this basis. The column vector basis for U_p can be written in the form

$$U_p = [u_1 \quad u_2 \quad \cdots \quad u_p]. \tag{2.4}$$

Now consider the matrix

$$V_m = [v_1 \quad v_2 \quad \cdots \quad v_m], \tag{2.5}$$

whose columns form a basis for the Krylov subspace $\mathcal{K}_p(K, m, U_p^{\perp})$ that is orthogonal to the computed invariant subspace U_p. We therefore write

$$[U_p \quad V_m]^H K[U_p \quad V_m] = \begin{bmatrix} U_p^HKU_p & U_p^HKV_m \\ 0 & V_m^HKV_m \end{bmatrix} = \begin{bmatrix} R_p & U_p^HKV_m \\ 0 & H_m \end{bmatrix}. \tag{2.6}$$

The matrix on the right-hand side, which we denote as $\hat{H}_{p+m}$, consists of a $p\times p$ block R_p which upper triangular (due to U_p being a Schur basis) and a $m\times m$ block H_m which is Hessenberg (due to V_m being an Arnoldi basis) The eigensystem of $\hat{H}_{p+m}$ is given by

$$\hat{H}_{p+m}\hat{X}_{p+m} = \hat{\Sigma}_{p+m}\hat{X}_{p+m} = \begin{bmatrix} \hat{\sigma}(R_p) & \\ & \hat{\sigma}(H_m) \end{bmatrix}\begin{bmatrix} X_p & \\ & X_m \end{bmatrix}, \tag{2.7}$$

where X_p, an upper triangular $p\times p$ matrix with unit diagonal represents the eigenvectors of R_p, and X_m which is a $m\times m$ matrix, represents the eigenvectors of H_m. In addition, $\hat{\sigma}(R_p)$ consists of the diagonal elements of R_p corresponding to the already converged eigenvalues, while $\hat{\sigma}(H_m)$ consists of the approximations to the remaining eigenvalues. Finally, the Ritz vectors Y_{p+m} of K can be computed from the relation

$$Y_{p+m} = [U_p \quad V_m]\,\hat{X}_{p+m} = [U_pX_p \quad V_mX_m], \tag{2.8}$$

with the last step following from (2.7). There is no need to transform the first p columns, since this will only reproduce the already converged eigenvectors. The remaining m columns contain approximations to the eigenvectors of K, and it is only necessary to transform those column vectors of X_m that correspond to converged eigenvectors. Fortunately, these converged eigenpairs can be identified, prior to performing this transformation, from the result

$$\|(K-\hat{\sigma}I)\,y_i\|_2 = \|v_{m+1}\|_2\,|e_m^Tx_i|, \tag{2.9}$$

so that the convergence test can be applied to the right-hand side in (2.9), without the need for explicitly evaluating all the Ritz vectors, in order to compute the eigenpair residuals on the right-hand side. This is very similar to the equivalent procedure used for detecting convergence in the symmetric Lanczos algorithm. Parenthetically, we note that it is important to compute the eigenpairs to high accuracy, since the use of reorthogonalization means that any errors will strongly affect the convergence and accuracy of subsequent eigenpairs.

The original invariant subspace U_p is enlarged by computing additional basis vectors from any newly converged eigenvectors by orthogonalization. However, each newly converged eigenvector is already orthogonal to the basis vectors of the original invariant subspace U_p, since from (2.8) such an eigenvector must be a linear combination of vectors in $U_p^{\perp}$. Therefore, it is only necessary to reorthogonalize the newly computed eigenvectors against each other in enlarging the basis for U_p.

The difficulty with the usual Arnoldi method is the indeterminate nature of its storage requirement, so that in practice a fixed number of iterations is carried out, and if a sufficient number of eigenpairs have not converged, then the method is reinitialized with a new starting vector. If

necessary, the value of the shift can also be changed at this point, without affecting the basis vectors of the invariant subspace U_p in (2.3), although the matrix R_p will change and must be recomputed.

After a sufficient number of eigenpairs have converged, the output from the algorithm will consist of the Schur matrices U_p and R_p. The matrix R_p will contain the eigenvalues of K in its diagonal entries, from which the eigenvalues of the original problem can be obtained using the transformation in (2.2). Frequently, only one or two of these eigenvalues are of interest, and an efficient method for computing the eigenvector corresponding to any specific eigenvalue is inverse iteration. For example, to compute the eigenvector corresponding to the diagonal element $R_{ii} = \hat{\sigma}$, we partition the matrix in the form

$$R_p = \begin{bmatrix} R_{11} & r & R_{12} \\ & \hat{\sigma} & s^T \\ & & R_{21} \end{bmatrix}, \tag{2.10}$$

where r and s are column vectors of length $(i-1)$ and $(p-i)$, respectively. With an accurate eigenvalue estimate, a single application of the inverse iteration procedure is sufficient to obtain a converged eigenvector. Therefore, we solve the small, dense, upper triangular system

$$(R_{11} - \hat{\sigma} I)\, y = -r, \tag{2.11}$$

from which the corresponding eigenvector w_i is obtained as

$$w_i = [u_1 \quad u_2 \quad \cdots \quad u_i] \begin{bmatrix} y \\ 1 \end{bmatrix}. \tag{2.12}$$

Note that the right-hand side of (2.12) is consistent with the partitioning of matrix R_p in (2.10).

3. ALGORITHM DESCRIPTION

A formal description of the algorithm is given below. The flop count estimates for the important steps in it are also given alongside in square brackets. This is followed by a brief discussion of some of the implementation details.

1. *Initialization*
   - starting shift λ.
   - eigenpair convergence tolerance *tol*.
   - number of desired eigenpairs *ntot*.
   - Krylov subspace dimension *mdim*, where *mdim* > *ntot*.
   - total number of inner Arnoldi iterations *maxits*.
   - maximum number of allowable consecutive inner Arnoldi iterations without a converged eigenpair *stagits*.
   - Set $p = 0$, $m = mdim$.

2. Assemble the matrices J and M. Choose a starting vector v_s.

3. Compute $J - \lambda M$, and compute its LU factorization, $[2N_t N_b + N_t N_b^2]$.

4. If $p \neq 0$, and the shift value has changed, recompute R_p the projection of K in the invariant subspace U_p, $[5pN_t N_b + 2p^2 N_t]$.

5. Restarted Iterations.

   (a) If $p \neq 0$, orthonormalize v_s against the computed Schur basis vectors $u_1, u_2, \ldots, u_p$. Then set $v_1 \leftarrow v_s$.

   (b) Inner Arnoldi iteration. For $j = 1$ to m do:

   i. Compute $\hat{v}_{j+1} = (J - \lambda M)^{-1} M v_j$, $[5mN_t N_b]$.
   ii. Project out the computed Schur basis vectors from $\hat{v}_{j+1}$, $[4mpN_t]$.
   iii. Compute $\hat{v}_{j+1} \leftarrow \hat{v}_{j+1} - \sum_{k=1}^{j} H_{k,j} v_k$, where $H_{k,j} = \hat{v}_{j+1}^H v_k$, $[4m^2 N_t]$.
   iv. Compute $\beta = H_{j+1,j} = \|\hat{v}_{j+1}\|$ and set $v_{j+1} \leftarrow \hat{v}_{j+1}/\beta$.
   v. End do.

   (c) Compute eigenvalues $\hat{\sigma}$ and eigenvectors x_i of the Hessenberg matrix H of order m using the EISPACK routine COMQR2, $[O(m^3)]$.

   (d) Check residuals and compute l, the number of additional eigenpairs that have converged in this inner Arnoldi iteration.

   (e) Backtransform the l converged eigenvectors x_i of H to obtain the corresponding Ritz vectors y_i of K, $[2mlN_t]$.

   (f) Enlarge the Schur basis set U_p and the projection matrix R_p from the l converged Ritz vectors, $[4l^2 N_t]$.

   (g) Set $p \leftarrow p + l$, $m \leftarrow m - l$. If $p \geq ntot$, exit to step 7.

   (h) If the number of consecutive inner iterations without converged eigenpairs is greater than *stagits*, exit to step 6.

   (i) If the total number of inner Arnoldi iterations is greater than *maxits*, exit to step 6.

   (j) Compute a new starting vector v, for the next inner Arnoldi iteration as a suitable linear combination of the unconverged Ritz vectors.

   (k) Go to step 5(a).

6. Compute a new shift value for the next set of computations.

7. If $p > ntot$, go to step 2. This test is made in the calling program to which the control is returned so that the default value for the new proposed shift can be modified to a user specified quantity, if so desired.

8. Identify the desired eigenvectors, and compute them by inverse iteration using the Schur matrices U_p and R_p, respectively.

9. End.

The matrix factorization in step 3, and the forward and back solves required in each invocation of step 5(b), are carried out using the ZGBFA/ZBGSL combination in LINPACK for banded complex matrices. The matrix J is overwritten by its LU factors and must therefore be saved or regenerated if step 2 is to be executed again with a different value of the shift λ. The matrix assembly is carried out in the calling program so that the module implementing the algorithm does not have to be modified for each individual application. In the inner Arnoldi iteration, much of the arithmetic involves the use of the level 1 BLAS subroutines ZDOTC and ZAXPY. These routines use loop unrolling and other optimizations to reduce the indexing overhead and promote register usage on scalar computers, and equivalent routines optimized for vector architectures will be quite efficient because they involve long vector operands that are accessed with unit stride.

In step 5(e), as mentioned earlier, only the converged eigenvectors found in 5(d) need to be backtransformed. In practise, however, it is also useful to obtain some of the nearly converged Ritz vectors in order to construct a good starting vector for the next outer loop. In step 5(f), the newly converged Ritz vectors are mutually orthogonalized and then added to the previous Schur basis, by overlaying the storage provided for the original Krylov basis set. In addition, by appropriately decrementing the size of the Krylov basis set in the next Arnoldi iteration, this overlaid storage is no longer required, so that the overall storage requirement of the procedure remains fixed.

In step 5(g), the criterion that is used to compute the starting vector is based on weighting each unconverged Ritz vector in the proportion of the product of the inverse of its residual, and the inverse of the distance of the corresponding approximate eigenvalue from the chosen shift. The first factor in this weighting is intended to further accelerate the convergence of the dominant eigenvalues, so that they may be quickly deflated out. The second factor, which is apparently arbitrary, is motivated by the observation that, particularly when the shift is poorly chosen, the restarted Arnoldi iteration can produce converged eigenvalues that are quite distant from the shift. This leads to two difficulties in the present context. First, these distant eigenvalues are possibly spurious, and it is clearly undesirable to be reorthogonalizing Krylov subspaces against spurious eigenvectors. Second, as shown below, the prospective shifts for the subsequent Arnoldi iterations are selected from among the "good" unconverged eigenvalues from some previous iterations, and any systematic procedure for doing this can be disrupted by extreme changes of the shift value in the complex plane. For these two reasons, it is best to strongly damp out the components of the Ritz vectors of these eigenvalues in the starting vector for the Arnoldi iteration, in this way to hopefully inhibit their convergence. Of course, other weighting functions can be used here, including those involving some positive power of this inverse separation and may prove to be even more effective in accomplishing the required damping.

Another aspect of the restarted Arnoldi iteration is that stagnation can occur with several iterations producing essentially no new converged eigenpairs. One situation when this potentially can arise is when a shift is located too close to an eigenvalue. This eigenvalue will of course converge rapidly, but its proximity to the shift will affect the subsequent progress of the Arnoldi iteration, so that even with reorthogonalization little information is obtained from these iterations to assist in the convergence of the subdominant eigenvalues. In step 5(h), if a check indicates that several successive iterations have produced no new converged eigenpairs, then further iteration is continued only if the residuals indicate that convergence is possibly imminent for some eigenpairs. However, if this situation persists, then selected "nearly converged" eigenvalues are placed in an internal buffer for use as possible future shift values, and the routine exits with a suggestion for a new shift value to be used in the next outer loop.

The buffer that is used to store the prospective shifts is maintained as a circular queue, and in step 6 the entry at the head of this queue is returned on output as the next proposed shift. By transferring the program control to the calling routine at this point, a decision can be made, based on any extraneous information, to either accept this new value, or to modify it to another alternative. The queue management strategy can be customized to the needs of the application. For example, a FIFO (first-in, first-out) queue will lead to the spectrum being mapped out in a radial fashion from the initial shift value. The most useful strategy for our purposes was to rank the prospective shifts by the decreasing magnitude of their imaginary parts, and to queue them accordingly, so that the shift selection is biased towards the region containing the stability-determining eigenvalues.

The possibility of stagnation in the shift selection procedure also cannot be overlooked, with several successive shifts being so close to each other that essentially no new information is obtained by using them. Note that each new shift value that is used requires that the matrix factorization be recomputed, and this is potentially the most expensive part of the computation. One strategy that we have used to avoid closely spaced shifts is to partition the region of the complex plane encircling the current shift in a number of radial sectors and to include only the nearest prospective shift values in each such sector into the queue. The use of an intermediate number of partitioning sectors is recommended here, in order to strike a balance between preventing stagnation on the one hand and inadvertently excluding some worthwhile directions for the movement of the shift, on the other hand.

The shift selection strategies described above are based on

the work of Cullum, Kerner, and Willoughby [2], who have proposed using "nearly converged" eigenvalues for this purpose as described above. Our modifications are intended to avoid closely spaced shifts and to bias the shift selection to the desired region of the complex plane. Their scheme can also be used as a default, if matrix factorizations are inexpensive and if location of the computed eigenvalues is of no particular consequence.

After the specified number of converged eigenvalues is obtained, the input to step 8 will consist of the partial Schur matrices U_p and R_p of K, computed at the most recently used value of the shift. A list of converged eigenvalues, that are backtransformed to that for the original unshifted problem is also returned. This list can be scanned and the eigenvector for any desired entry in it can be computed by inverse iteration. This requires the solution of a small dense, complex, upper triangular matrix system, and this is carried out using a modified version of the LINPACK routine ZTRSL.

The choice of the band matrix format is not essential to the algorithm, and with trivial modifications to the program, any other alternative sparse matrix format may be used, as long as suitable routines are provided for performing the matrix factorization, triangular solves, and matrix-vector multiplications. In particular, for large-scale "out-of-core" applications, the LINPACK band matrix routines will not be efficient without further modification. A good alternative in this case is the frontal method, which is more economical in its core storage and I/O requirements. Here, the assembly of the matrix $J-\lambda M$ is overlapped with the factorization. The same strategy can be used to compute the matrix-vector product involving M without ever explicitly assembling this matrix (specifically, when this assembly would require random access to a large array stored on disk), by using an element-by-element technique with the element matrices being regenerated whenever required. This will incur some additional computation, but should be economical unless an unusally large number of matrix-vector multiplications is performed.

4. TEST PROBLEMS AND EXPERIMENTAL RESULTS

In this section, we describe two applications that were specifically programmed to evaluate the correctness and efficiency of the algorithm. The discretization methods used are standard, but specific to this work, and we give all the relevant details for the sake of completeness. All numerical results and timings were obtained on a SUN 4/280 workstation using an f77 FORTRAN compiler with full optimization.

A. *Orr–Sommerfeld Equation*

The Orr–Sommerfeld equation describes the stability of small two-dimensional perturbations of the form

$$\phi(x)\exp[i\alpha(z-\sigma t)] \tag{4.1}$$

to the stream function of a base shear flow $U(x)$ between two parallel plates of infinite extent located at $x=\pm 1$, respectively. This equation, which is relevant to the onset and development of turbulence in channel flows [3], is given by

$$\frac{i}{\alpha R}[\phi'''' - 2\alpha^2\phi'' + \alpha^4\phi] + [(U-\sigma)(\phi''-\alpha^2\phi) - U''\phi] = 0, \tag{4.2}$$

$$\phi = \phi' = 0, \qquad \text{at} \quad x = \pm 1. \tag{4.3}$$

Here R denotes the Reynolds number, and α and σ denote the streamwise wavenumber and the complex wave speed of the perturbation. The unstable waves are those with $\mathrm{Im}(\sigma) > 0$, and this definition is different from that given in Section 1, but is used here to maintain consistency with the work of previous investigators on this problem. The two relevant base velocity profiles are the plane Poiseuille flow U_p, and the plane Couette flow U_c, which are respectively given by

$$U_p = 1 - x^2, \qquad U_c = x. \tag{4.4}$$

The weak form of (4.2) is obtained by taking test functions $w \in \mathcal{H}_0^2(-1, +1)$ to obtain, after integration by parts,

$$\int_{-1}^{+1}\left[\frac{i}{\alpha R}(w''\phi'' + 2\alpha^2 w'\phi'' + \alpha^4 w\phi) - \{(U-\sigma)(w'\phi' + \alpha^2 w\phi) + U'w\phi' + U''w\phi\}\right]dx = 0. \tag{4.5}$$

The mesh used in the discretization consists of the interior nodal points $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{N-2}\}$, and ϕ is expanded using Hermite cubic basis functions,

$$\phi = \sum_{j=1}^{N-2} \hat{\phi}_j \Psi_j + \sum_{j=1}^{N-2} \hat{\phi}'_j \omega_j \tag{4.6}$$

where $\{\hat{\phi}_j, \hat{\phi}'_j\}$ denote the nodal values and the derivatives, respectively, at the point $\hat{x}_j$. Since boundary layers will develop at the walls and along the centerline for large values of R, the stretching transformation

$$\hat{x}_j = \frac{x_j}{|x_j|}\sin^2(\pi x_j/2), \tag{4.7}$$

where x_j is an equally-spaced partition of the interval $(-1, +1)$, is used to cluster nodes in the regions where large spatial gradients in the eigenvectors are expected.

The use of Galerkin's method with test functions w given by the set $\{\Psi_i, \omega_i\}$, followed by the evaluation of the various integrals in (4.5) by numerical quadrature on each element, leads to a matrix eigenvalue problem of the form (1.8), with matrices J and M of order $2N$.

Orszag [9] has given a list of the 30 least stable eigenvalues for the case $\alpha = 1$, Re = 10,000, for perturbations to a base Poiseuille flow, which were computed using a spectral discretization in conjunction with the well-known QR algorithm (which is implemented in EISPACK). His results are shown plotted here in Fig. 1a. We briefly remark on the eigenvalue distribution for this problem. First, the real parts of all the eigenvalues lie in the interval (0.0, 1.0). Second, the eigenvalues with large negative imaginary parts all have their corresponding real parts clustered around a value of $\frac{2}{3}$, which is the mean fluid velocity in the channel. For somewhat larger values of the $\mathrm{Im}(\sigma)$, there are two branches in the spectrum, consisting of the "slow" modes with $\mathrm{Re}(\sigma) < \frac{2}{3}$, and the "fast" modes with $\mathrm{Re}(\sigma) > \frac{2}{3}$. This structure can be seen in Fig. 1a. The eigenvalues of the slow modes are somewhat scattered, while those of the fast modes are seen to roughly lie on a straight line, inclined to the right from the vertical. The fast eigenvalues are in fact nearly degenerate (to the first few decimal places) and each corresponding point in Fig. 1a actually denotes a pair of eigenvalues, whose eigenvectors are respectively symmetric and antisymmetric relative to the channel centerline. Orszag [9] apparently did not compute eigenvectors, but used the fact that the symmetric and antisymmetric modes decouple

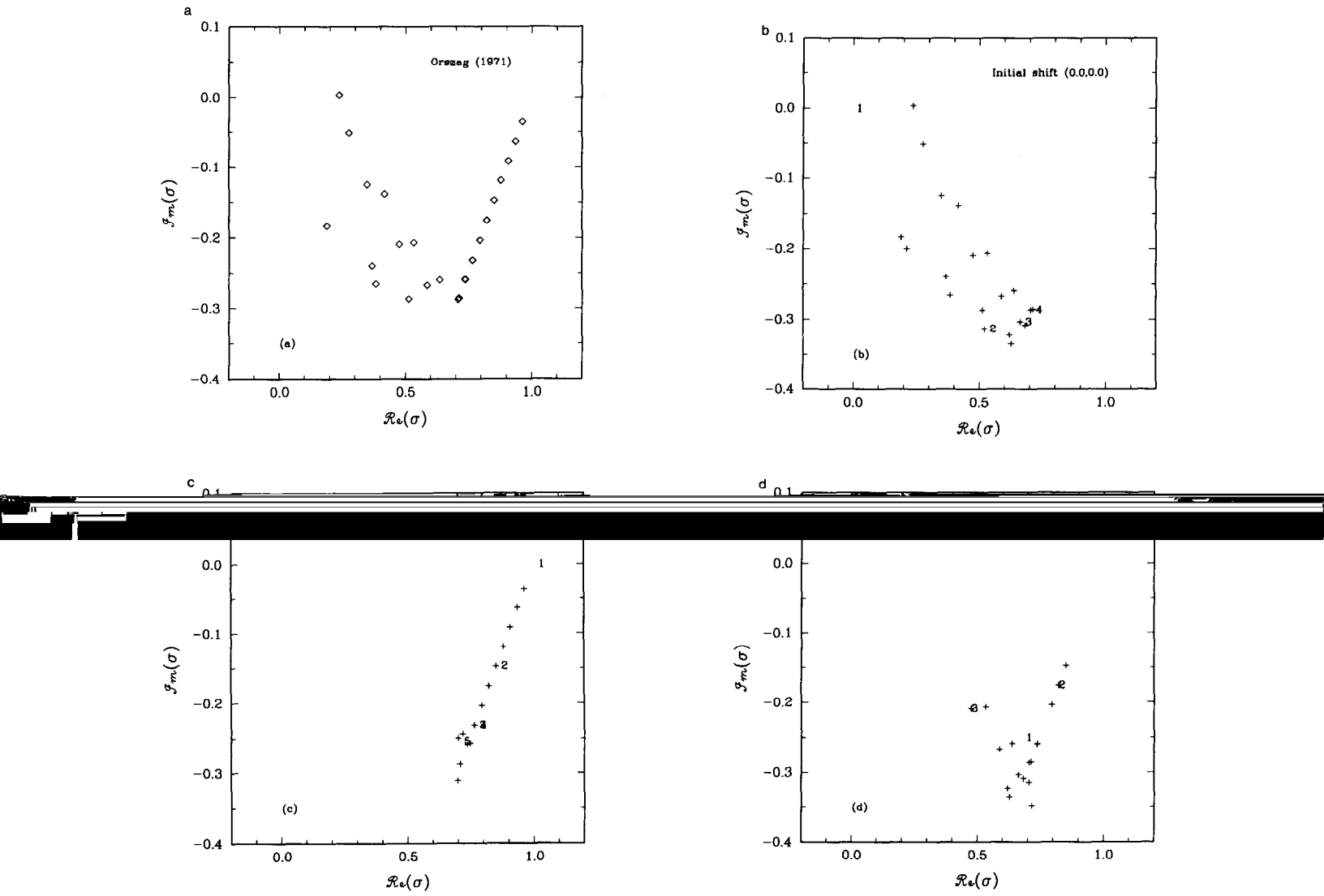

FIG. 1. Partial eigenvalue spectrum for the Orr–Sommerfeld quation for $\alpha = 1$, Re = 10,000: (a) 30 least stable eigenvalues reported by Orszag [9]. (b), (c), and (d) are the first 20 eigenvalues obtained from various different starting values with the present algorithm.

TABLE I

Convergence History of the Computations for the Orr–Sommerfeld Equation

Shift value	No. of inner iteration	No. of computed eigenvalues	CPU seconds
(0.0, 0.0)	12	10	128.9
(0.523, −0.314)	12	7	67.8
(0.663, −0.303)	12	2	36.5
(0.705, −0.281)	12	2	33.0

in the original problem to obtain and identify these eigenvalues from separate computations.

Our computational results were obtained using a mesh with $N = 201$, with the following values for the various parameters in the algorithm, $mdim = 30$, $ntot = 20$, $maxits = 12$, $stagits = 5$. In Fig. 1b, the eigenvalues that were computed starting from an initial shift (0.0, 0.0) are shown, and the convergence history is summarized in Table I. A few remarks are necessary to place the entries of Table I in context. First, the time taken for the LU factorization is quite negligible for the small bandwidth matrix obtained in this problem. Second, the dominant contribution to the CPU time in the inner Arnoldi iteration is from the EISPACK routine COMQR2. This is reflected in the fact that the CPU time per iteration decreases with the size of the Arnoldi subspace, as more eigenvalues are computed. The decrease in the size of this subspace also explains why more iterations are required in the later stages of this algorithm in order to compute each additional eigenvalue. Note, however, that the combination of these two effects leads to the CPU time for obtaining each additional eigenvalue being quite uniform over the progress of the computation. Third, the computations that were performed with the third shift value had to be prematurely terminated after 10 inner Arnoldi iterations because of a perceived stagnation.

There is excellent agreement (to several decimal places of accuracy) of our computed eigenvalues with the earlier results of Orszag [9]. In fact, we have found that one of our computed eigenvalues, located roughly at (0.21, −0.199) in Fig. 1b, to be inadvertently missing from the table of values reported by Orzag in his paper. The various numerals plotted in Figs. 1a–1d correspond to the locations of the shift values that were automatically generated by the routine, using a priority scheme in which prospective shift values were ranked by decreasing imaginary parts. We note that in spite of this heuristic, the algorithm was unable to uncover any of the fast modes, even though the direction of the movement of the shift values indicates that this might have happened if the algorithm had been allowed to compute a much larger number of eigenvalues.

In order to obtain the fast eigenvalues (whose existence was of course known a priori, but in fact could have easily been ascertained by performing computations at smaller of R and extrapolating), the program was restarted with an initial shift (1.0, 0.0) and the results from this are shown in Fig. 1c. As mentioned earlier, each point in this figure actually corresponds to a pair of nearly degenerate eigenvalues. As a final check of the shift selection strategy, a starting shift of (0.67, −0.25) was used and the results are shown in Fig. 1d. Again, the movement of the shifts is seen to be towards uncovering the least stable eigenvalue, although in this case, insufficient eigenvalues were computed for this strategy to proceed to completion. The eigenvalues that were obtained starting from this initial shift consist of a few fast modes and a few slow modes.

The real and imaginary parts of the eigenvectors corre-

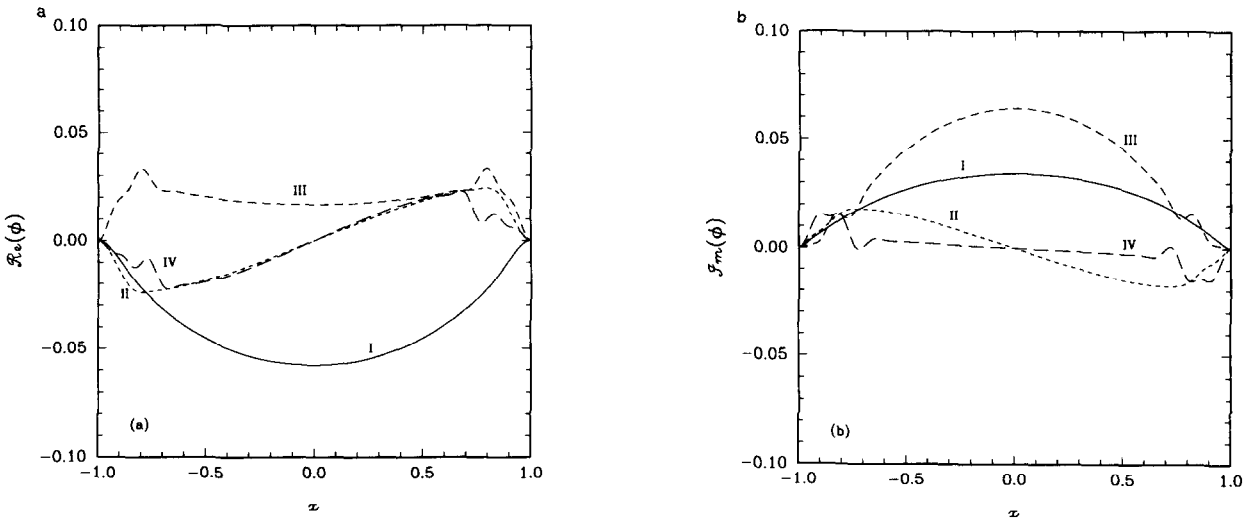

FIG. 2. Real and imaginary parts of the eigenvectors for the four least stable modes in Fig. 1b: I. (0.237, 3.74×10^{-3}); II. (0.277, -5.09×10^{-2}); III. (0.349, −0.124), IV. (0.416, −0.138).

sponding to the four least stable eigenvalues in Fig. 1a are shown in Figs. 2a and 2b, respectively. The mode denoted I in 2a and b is slightly unstable, and this, along with the mode denoted III, is seen to be symmetric about the channel centerline. The modes II and IV, on the other hand, are seen to be antisymmetric. The slow modes are spatially uniform over much of the interior of the channel, but show boundary layer-like behavior at the walls, with the corresponding variations being rather more pronounced for the higher modes. Similarly, the eigenvectors corresponding to the four least stable fast modes are shown in Fig. 3a and b. Here, modes I and IV are seen to be antisymmetric, while modes II and III are symmetric. These modes are spatially uniform in the wall region but show a critical layer-like behavior in the region close to the channel center line.

B. *Chemical Reactor Equations*

The time-dependent behavior of a non-adiabatic tubular reactor is described by the following equations given in dimensionless form [8]

$$\frac{\partial y_1}{\partial t} = \frac{1}{Pe_1}\frac{\partial^2 y_1}{\partial x^2} - \frac{\partial y_1}{\partial x} - g_1(y_1, y_2), \tag{4.8}$$

$$Le\frac{\partial y_2}{\partial t} = \frac{1}{Pe_2}\frac{\partial^2 y_2}{\partial x^2} - \frac{\partial y_2}{\partial x} - g_2(y_1, y_2), \tag{4.9}$$

where y_1 and y_2 are the dimensionless reactant conversion and temperature, respectively. The functions g_1 for a temperature-dependent, first-order reaction, and g_2 for the combination of the heat transfer through the walls and th heat of reaction, are respectively given by

$$g_1 = Day_1 \exp\left(\gamma + \frac{\gamma}{y_2}\right), \tag{4.10}$$

$$g_2 = \beta(y_2 - 1) - BDay_1 \exp\left(\gamma + \frac{\gamma}{y_2}\right). \tag{4.11}$$

At the reactor inlet and exit, y_1 and y_2 are assumed to satisf Dankwerts' boundary conditions, given by

$$x = 0; \qquad \frac{\partial y_1}{\partial x} = Pe_1(y_1 - 1), \qquad \frac{\partial y_2}{\partial x} = Pe_2(y_2 - 1), \tag{4.12}$$

$$x = 1; \qquad \frac{\partial y_1}{\partial x} = 0, \qquad \frac{\partial y_2}{\partial x} = 0, \tag{4.13}$$

The various parameters appearing in the model are the Peclet numbers for heat and mass transport Pe_1 and Pe_2, the Lewis number Le, the Damkohler number Da, the dimensionless activation energy γ, the dimensionless heat of reaction B, and the dimensionless heat transfer coefficient β. Some simplifying assumptions are made in the most general model, such as taking the reactor inlet and wall temperatures to be equal, which allow us to compare our results with those of Roose [11].

The discretization of (4.8)–(4.9) is carried out using piecewise continuous biquadratic basis functions $\Phi_i(x)$, defined on 3-node elements, and Galerkin's method is used

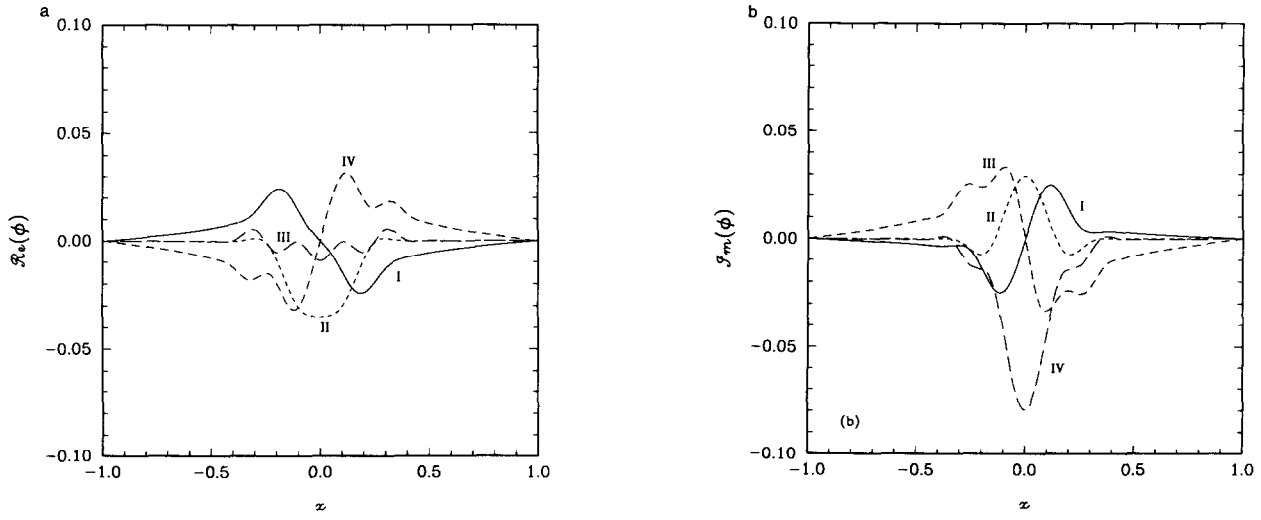

FIG. 3. Real and Imaginary parts of the eigenvectors for the four least stable modes in Fig. 1c: I. (0.9646, -3.52×10^{-2}); II. (0.965, -3.52×10^{-2}); III. (0.936, -6.32×10^{-2}); IV. (0.936, -6.32×10^{-2}).

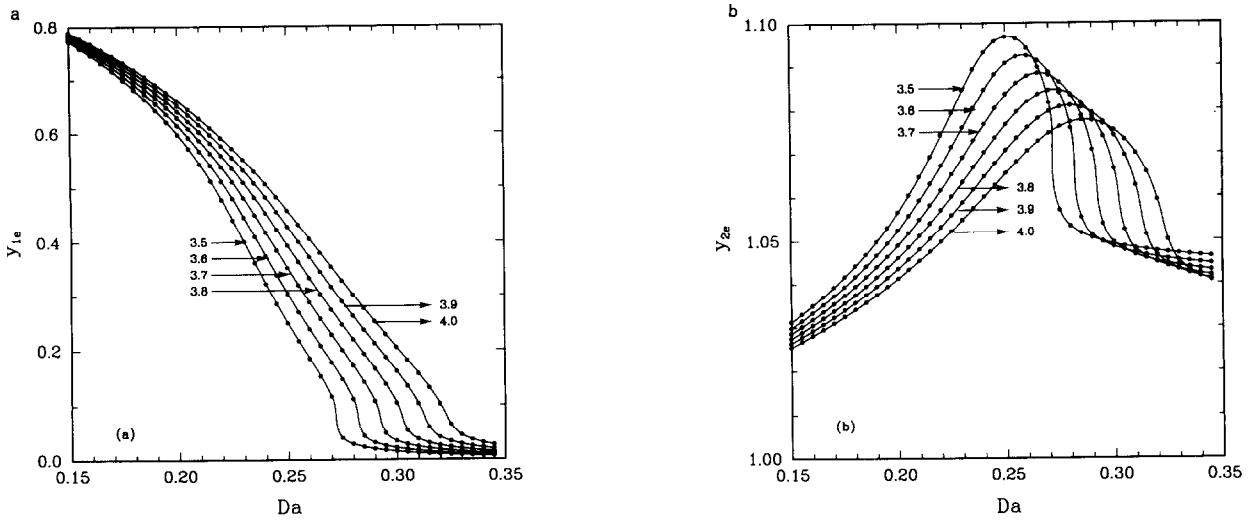

FIG. 4. Exit values of y_1 and y_2 obtained by solving for the stationary solutions of (4.8)–(4.13). The problem parameters are $Pe_1 = Pe_2 = 5.0$, $\gamma = 25.0$, $\beta = 0.5$.

to obtain the numerical discretization from the weak form of (4.8) and (4.9). The details of the method used to solve for the stationary equations and to compute the eigenvalues of the perturbation equations closely follow the earlier description in Section 1, and these details are therefore omitted here for the sake of brevity.

Our results were obtained for the following values of the problem parameters, $Pe_1 = Pe_2 = 5.0$, $\gamma = 25.0$, $B = 0.5$. The value of β was varied in the range (3.5, 4.0), and, at each value of β, solutions were computed using Da as a continuation parameter. In Fig. 4, the reactor exit values are shown plotted for values of Da between 0.15 and 0.35. The curve for $\beta = 3.5$, in particular, shows a tendency towards forming a "neck" in the region where two limit points begin to appear. As a result, the simple continuation procedure breaks down for $\beta < 3.5$, with multiple solutions appearing for intermediate values of Da between these two limit points. Alternative methods based on arc-length continuation could have been used to extend the computation to values of $\beta < 3.5$, but this was not pursued here.

For each value of the continuation parameter Da, after computing a converged stationary solution, the Arnoldi routine was used to obtain a few of the most unstable eigenvalues of the discretized perturbation equations. The various parameters for the Arnoldi routine were $mdim = 20$, $ntot = 12$, $tol = 10^{-10}$, $maxits = 8$, $stagits = 5$. In this problem, a good shift value can always be estimated from the preceding computations, and a single shift is often

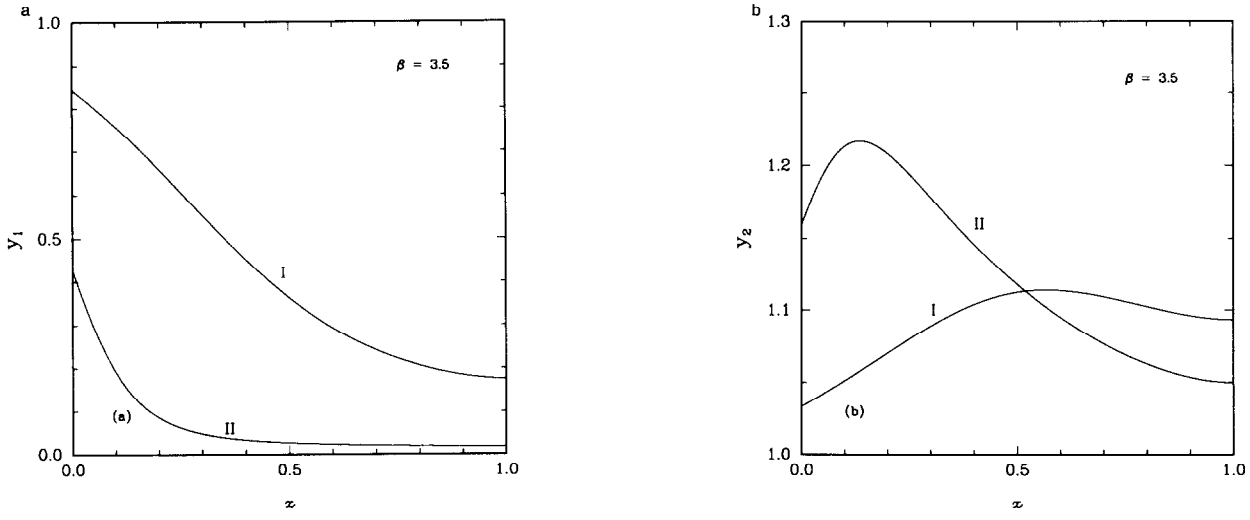

FIG. 5. The spatial form of the stationary solutions for (a) y_1 and (b) y_2, at the lower and upper Hopf bifurcation points (denoted by I and II, respectively).

a 0.4

b 0.4

enough to compute all the required eigenvalues. For the parameter values used in the present problem, each Newton iteration for computing the steady solution took about 0.45 CPU seconds, with typically 4–5 iterations for the required convergence from a good starting guess. The computation of the 12 leading eigenvalues and two leading eigenvectors took about 53 CPU seconds. Here the first and second Arnoldi iterations required 13 and 8 CPU seconds, respectively, while each subsequent iteration required about 5 CPU seconds (the fraction of the CPU time in the matrix factorization and the eigenvector computation was quite negligible). In this example, 10 of the 12 computed eigenvalues were typically obtained in the first three inner Arnoldi iterations. This observation could have been used to reduce the computational costs by over a factor of two without a great loss of information from the point of view of the application.

By monitoring the least stable eigenvalue during the parameter continuation with Da, it was possible to obtain a coarse estimate of the interval containing a bifurcation. This coarse estimate was subsequently refined using interval bisection in the usual way (this procedure can be easily automated for a "production" application). In this way, the two Hopf bifurcation points, termed Da_l and Da_u were computed for each value of β in the range (3.5, 3.8). These values along with the magnitude of the imaginary part of the bifurcation eigenvalues are listed in Table II. Detailed calculations for other values of β were not pursued.

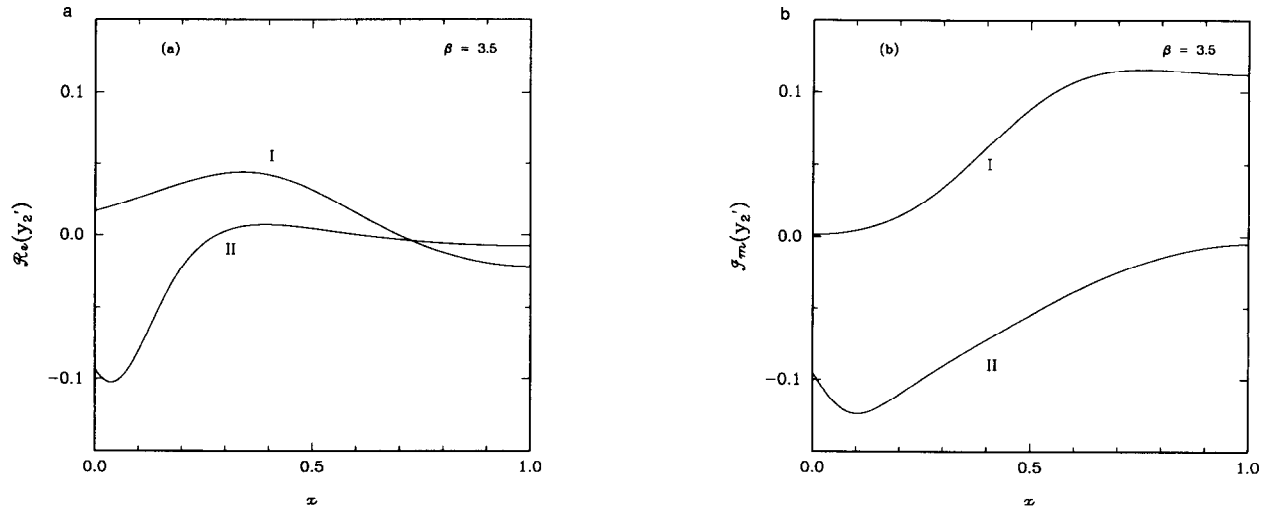

FIG. 7. The spatial form of the (a) real and (b) imaginary parts of the unstable eigenvector perturbation to y_2 at the lower and upper Hopf points (denoted by I and II, respectively).

TABLE II

Values of the Critical Damkohler Number at the Upper and Lower Hopf Points for Various β, $Pe_1 = Pe_2 = 5.0$, $\gamma = 25.0$, $B = 0.5$

	Lower Hopf point		Upper Hopf point	
β	Da_l	$\mathrm{Im}(\sigma_l)$	Da_u	$\mathrm{Im}(\sigma_u)$
3.5	0.2164	1.552	0.2952	5.38
3.6	0.2701	1.599	0.3078	5.53
3.7	0.2789	1.645	0.3206	5.68
3.8	0.2876	1.698	0.3337	5.84

In Fig. 5, we show the stationary reactor profiles for the conversion and temperature in the case $\beta = 3.5$ at the two Hopf bifurcation points. These profiles are fairly uniform at Da_l, but for Da_u, steep variations are found in the entrance region of the reactor where most of the reactant undergoes rapid conversion. The real and imaginary parts of the bifurcating eigenvectors are shown in Fig. 6 for the conversion, and in Fig. 7 for the temperature. Since the bifurcating solution in this case will appear in the form of a standing time-dependent oscillation, the shape of the modes giving an idea of the location in the reactor where these oscillations might be expected to have the largest relative amplitude. The absolute magnitude of the amplitudes of the bifurcating solutions, however, requires a further nonlinear analysis, and cannot be obtained from the present approach alone.

5. SUMMARY

A procedure for obtaining the partial eigenspectrum of large sparse matrix eigenvalue problems arising in finite element stability applications has been described. The underlying algorithm is that of Saad [12] but has been modified in some ways to tailor the implementation for this set of applications. The method has been used to study two test problems, which are apparently small, but whose solution using standard eigenvalue software would have yielded the present results much less economically. This performance gap will increase dramatically for larger problems, more so, in the context of an implementation on a vector or parallel computer.

In the earlier related work, there has been an emphasis on developing methods for "filtering" the starting vector for Arnoldi's method (or any other equivalent Krylov subspace based eigenvalue algorithm) so that this starting vector is rich in the component of the most desired eigenvector. The algorithms that have been proposed for performing this filtering include for example, a time-integration of the transient version of the eigenvalue problem [1, 4], and Chebyshev preconditioning [12, 6]. Although we have not experimented extensively with either technique, the former requires the specification of various parameters such as the integration step size, integration time span, etc., which must be separately estimated for each application. More importantly, however, the effectiveness of these various filtering schemes is diminished by the fact that further amplification of this eigenvector is not necessarily favored in Arnoldi's method, which can in fact undo all the work invested in the filtering in the first place. The approach taken in this paper, on the other hand, has been to emphasize the use of a shift selection technique in order to force the convergence of the desired eigenvalues.

From our experiments, we have concluded that by reorthogonalizing the Arnoldi subspace against the computed invariant subspace, an effective and stable way of computing a reasonable number of eigenvalues is obtained, without the appearance of "duplicates" and without an excessive number of matrix factorizations. The use of this scheme for computing a large number of eigenvalues can lead to some difficulty, since any error in the computation of the first few eigenpairs will affect the accuracy and convergence of all subsequent eigenpairs. It is possible, therefore, that because of this limitation, the most desired eigenvalues in a given application cannot be extracted from a particular starting shift. A simple remedy is to reinitialize the program with a different starting shift. We have described some heuristics for the shift selection that can reduce the possibility of this happening for our applications, but a fully satisfactory automatic procedure is difficult to obtain in any scheme that attempts to infer the global distribution of the eigenvalues from the computation of a small subset. In practice, however, a rough idea of the eigenvalue distribution can be inferred by extrapolating from a computation on either a coarse mesh or at a nearby set of problem parameter values. Furthermore, this extrapolation does not have to be particularly accurate, since the present scheme will always compute a reasonable number of eigenvalues at each step, and doing this in conjunction with the automatic shift selection procedure will provide a good assurance that any "overtaking" modes do not go inadvertently unidentified.

ACKNOWLEDGMENTS

The author is indebted to the reviewers for their helpful comments.

REFERENCES

1. K. N. Christodolou and L. E. Scriven, University of Minnesota Report No. UMSI 88/130, 1988 (unpublished).
2. J. Cullum, W. Kerner, and R. A. Willoughby, *A Generalized Nonsymmetric Lanczos Procedure*, Proceedings, Workshop on Practical Iterative Methods for Large Scale Computation, University of Minnesota, 1988 (unpublished).

3. P. G. Drazin and W. H. Reid, *Hydrodynamic Stability* (Cambridge Univ. Press, New York, 1981).
4. I. Goldhirsch, S. A. Orszag, and B. K. Maulik, *J. Sci. Comput.* **2**, 33 (1987).
5. D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods* (SIAM, Philadelphia, 1977).
6. D. Ho and F. Chatelin, private communication (1988).
7. C. P. Jackson, *J. Fluid Mech.* **182**, 23 (1987).
8. K. F. Jensen and W. H. Ray, *Chem. Eng. Sci.* **37**, 199 (1982).
9. S. A. Orszag, *J. Fluid Mech.* **182**, 23 (1987).
10. B. Parlett and Y. Saad, *Linear Algebra Appl.* **88**, 575 (1987).
11. D. Roose, *J. Comput. Appl. Math.* **12**, 517 (1985).
12. Y. Saad, *Numerical Solution of Large Nonsymmetric Eigenvalue Problems*, Proceedings, Workshop on Practical Iterative Methods for Large Scale Computation, University of Minnesota, 1988 (unpublished).
13. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method* (Prentice–Hall, Englewood Cliffs, NJ, 1973).
14. J. Wilkinson, *The Algebraic Eigenvalue Problem* (Oxford Univ. Press, New York, 1988).