Implicitly restarted Arnoldi methods and eigenvalues of the discretized Navier Stokes equations.

R. B. Lehoucq[*] J. A. Scott[†]

October 16, 1997

Abstract

We are concerned with finding a few eigenvalues of the large sparse nonsymmetric generalized eigenvalue problem $Ax = \lambda Bx$ that arises in stability studies of incompressible fluid flow. The matrices have a block structure that is typical of mixed finite-element discretizations for such problems. We examine the use of shift-invert and Cayley transformations in conjunction with the implicitly restarted Arnoldi method along with using a semi-inner product induced by B and purification techniques. Numerical results are presented for some model problems arising from the ENTWIFE finite-element package. Our conclusion is that, with careful implementation, implicitly restarted Arnoldi methods are reliable for linear stability analysis.

AMS classification: Primary 65F15; Secondary 65F50

Key Words: eigenvalues, sparse nonsymmetric matrices, Arnoldi's method.

1 Introduction

Mixed finite-element discretizations of time-dependent equations modelling incompressible fluid flow problems typically produce nonlinear finite dimen-

[*] Sandia National Laboratories, MS 1110, P.O.Box 5800 Albuquerque, NM 87185-1110, rlehoucq@cs.sandia.gov. This work was funded by the Applied Mathematical Sciences program, U.S. DOE, Office of Energy Research, and undertaken at Sandia National Labs, operated for DOE under contract No. DE-AL04-94AL8500.

[†] Computing and Information Systems Department, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England. sct@letterbox.rl.ac.uk

sional systems of the form

$$\begin{aligned} M\dot{u} + H(u)u + Lu + Cp &= b, \\ C^T u &= c, \end{aligned} \tag{1.1}$$

where $u \in \mathbf{R}^n$, $p \in \mathbf{R}^m$ with $n > m$. M and L are symmetric positive definite $n \times n$ matrices, $H(u)$ is a nonsymmetric $n \times n$ matrix, and C is an $n \times m$ matrix of full rank. Linearized stability analysis of (1.1) leads to the problem of finding a few eigenvalues of the generalized eigenvalue problem

$$Ax = \lambda Bx \tag{1.2}$$

where

$$A = \begin{pmatrix} K & C \\ C^T & 0 \end{pmatrix}, \quad B = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad x = \begin{pmatrix} u \\ p \end{pmatrix}. \tag{1.3}$$

In the case of the so-called primative variable formulation of the discretized Navier Stokes equations for incompressible flow, u and p denote the velocity and pressure degrees of freedom, respectively (see, for example, Cliffe, Garratt and Spence, 1993). The matrix M is the mass matrix and K is nonsymmetric because of the linearization of the convection term. The matrices K, C, and M are all sparse and in real applications are very large.

For stability analysis, the interest lies in computing the eigenvalues of smallest real part (the left-most eigenvalues) (Georgescu, 1985 and Sattinger, 1973). Of special interest is the case when the eigenvalues of smallest real part are complex, because algorithms for the detection of Hopf bifurcations in parameter dependent systems can be developed from knowledge of these eigenvalues. A complication that arises is that the eigenvalue problem can have infinite eigenvalues, corresponding to eigenvectors of the form $\begin{pmatrix} 0 \\ p \end{pmatrix}$ (see Malkus, 1981, Ericsson, 1986 and Cliffe, Garratt and Spence, 1994).

A standard approach for finding the left-most eigenvalues of the discretized Navier Stokes equations is to use a rational transformation such as a shift-invert or a Cayley transformation, and then to apply an iterative technique, for example, subspace iteration or Arnoldi's method, to the transformed problem. Traditionally, subspace iteration has been the method of choice because Arnoldi's method was perceived as being less reliable. Experiments reported by Garratt (1991) found that Arnoldi's method sometimes missed the sought-after left-most eigenvalue and, because reliability is more important than efficiency in linear stability analysis, Garratt favoured subspace iteration. However, the recent work of Meerbergen and Spence (1997) demonstrates, in theory, that the implicitly restarted Arnoldi method of Sorensen (1992), combined with shift-invert transformations, can be successfully employed to compute the left-most eigenvalues of generalized eigenvalue problems with the block structure (1.3). In this report, we look at using the package ARPACK of Lehoucq, Sorensen and Yang (1998), which implements the implicitly restarted Arnoldi method, to compute eigenvalues of the discretized Navier Stokes equations. Our results show that implicitly restarted Arnoldi methods can be used reliably.

The outline of this report is as follows. In Sections 2 and 3 we give brief introductions to the implicitly restarted Arnoldi method and matrix transformations for generalized eigenvalues. We then look at shift-invert and Cayley transformations in Sections 4 and 5. We discuss the generalized Cayley and modified Cayley transformations; the latter was introduced by Cliffe et al. (1993) as a way of mapping the unwanted infinite eigenvalues to a part of the spectrum where they are unlikely to be computed by the eigensolver. The algorithm which we propose for computing eigenvalues of the discretized Navier Stokes equations is outlined in Section 6. In Section 7, we explain our choice of linear equation solver. Section 8 introduces the software package `ARPACK`, that implements the implicitly restarted Arnoldi method. We also highlight the modifications to `ARPACK` that allowed us to use Cayley transformations. Numerical results are presented in Section 9. Finally, we make some comments on our findings and on possible future work.

Throughout this report, the finite eigenvalues of the generalized eigenvalue problem (1.2) are denoted by λ_i $(i = 1, \ldots, n - m)$ and it is assumed that they are ordered by increasing real parts i.e. $i > j \Rightarrow \mathrm{Re}(\lambda_i) \geq \mathrm{Re}(\lambda_j)$. The standard inner product of two vectors x and y is $x^H y$ where x^H is the complex conjugate transpose of x. (If the vector x is real then $x^H = x^T$.) The Euclidean norm of a vector x is defined to be $\|x\|_2 = \sqrt{x^H x}$. The B semi-inner product of two vectors x and y is $x^H B y$ and induces the B semi-norm $\|x\|_B = \sqrt{x^H B x}$. The use of $\|x\|$ implies that either the standard or B semi-norm may be used.

2 The implicitly restarted Arnoldi method

A relatively recent variant of Arnoldi's method is that developed by Sorensen (1992) as a more efficient and numerically stable way to implement restarting. The scheme is called *implicit* because the starting vector is updated by combining the implicitly shifted QR algorithm with an Arnoldi reduction to obtain a truncated form of the implicitly shifted QR iteration. One of the benefits of an implicitly restarted Arnoldi method is that it avoids the need to restart the Arnoldi reduction from scratch and thus fixes storage requirements.

Consider an Arnoldi reduction of length r of a matrix A

$$AV_r = V_r H_r + f_r e_r^T, \tag{2.1}$$

where the $n \times r$ matrix V_r has orthogonal columns, $V_r^H f_r = 0$, and H_r is an $r \times r$ upper Hessenberg matrix. The columns of V_r are an orthogonal basis for the Krylov space $\mathcal{K}_r(A, v_1) = \{v_1, Av_1, \cdots, A^{r-1}v_1\}$, where v_1 is the first column of V_r. Let $\psi_i(\lambda) = \prod_{j=1}^{i}(\lambda - \gamma_j)$ where $i \le r$. In the discussion that follows, we will show how to implicitly compute an orthogonal basis (and corresponding Arnoldi reduction) for the updated Krylov space $\psi_i(A)\mathcal{K}_{r-i}(A, v_1)$, that is, we compute an orthogonal basis for the updated space without using A. In particular, if all the γ_j are equal to zero, then we have equivalently performed subspace iteration on $\mathcal{K}_{r-i}(A, v_1)$.

A tedious but straightforward induction argument shows that

$$\psi_i(A)V_{r-i} = V_r G_{r-i}, \tag{2.2}$$

where G_{r-i} contains the leading $r-i$ columns of $\psi_i(H_r)$. If the QR factorization of G_{r-i} is $U_1 R_1$, where R_1 is an upper triangular matrix of order $r-i$, then

$$\psi(A)V_{r-i} = V_r U_1 R_1 \equiv V_{r-i}^+ R_1. \tag{2.3}$$

Thus the $r-i$ columns of V_{r-i}^+ provide an orthogonal basis for the range of $\psi(A)V_{r-i}$. In particular, the starting vector v_1 associated with the Arnoldi reduction (2.1) has been updated with the polynomial filter $\psi_i(\lambda)$. The *roots* or *implicit shifts* γ_j may be selected to filter unwanted information from the starting vector and hence from the Arnoldi reduction.

Although another Arnoldi reduction can be computed using the first column of V_{r-i}^+ as the starting vector, the Arnoldi reduction (2.1) can be updated directly to obtain

$$AV_{r-i}^+ = V_{r-i}^+ H_{r-i}^+ + f_{r-i}^+ e_{r-i}^T. \tag{2.4}$$

We now briefly explain how this can be accomplished.

1. Perform i steps of the implicitly shifted QR algorithm on H_r. This results in the similarity transformation $H_rU = UH_r^+$, where U is an orthogonal matrix of order r and H_r^+ is also upper Hessenberg.
2. A classical result shows that the first $r-i$ columns of U are equal to U_1 and, therefore, they provide an orthogonal basis for the leading $r-i$ columns of $\psi_i(H_r)$. (See Stewart, 1973, pages 351–355 or Watkins, 1991, pages 293–305.)
3. Postmultiply the Arnoldi reduction (2.1) with U_1 to obtain $AV_rU_1 = V_rH_rU_1 + f_re_r^TU_1$.
4. Equate the first $r-i$ columns of $H_rU = UH_r^+$ to get $H_rU_1 = U_1H_{r-i}^+ + \beta_{r-i}^+u_{r-i+1}$, where H_{r-i}^+ is the leading principal matrix of order $r-i$ in H_r^+, β_{r-i}^+ is the subdiagonal element in row $r-i$ of H_r^+, and u_{r-i+1} is column $r-i+1$ of U.
5. Insert the expression for H_rU_1 derived in step 4 into the postmultiplied Arnoldi reduction of step 3 to obtain (2.4).

The above development shows that the IRAM is subspace iteration (via $\psi(A)$) in disguise. This equivalence was mentioned in Meerbergen and Spence (1997). See Lehoucq (1997) for further details on the connection between subspace iteration and the implicitly shifted QR algorithm.

We end our discussion by addressing the issue of the selection of the implicit shifts γ_j. One possible shift selection strategy is the so-called *exact* shift (Sorensen 1992) strategy, where the r eigenvalues of H_r are partitioned into a set of k wanted and l unwanted ones according to a selection criterion. For example, if the left-most eigenvalues are sought, the unwanted set comprises the l right-most eigenvalues. The unwanted eigenvalues are used as the shifts and thus a polynomial filter of degree $i = l$ is applied. This is equivalent to restarting the Arnoldi reduction with a linear combination of the approximate eigenvectors associated with the wanted eigenvalues. Another possibility is to use implicit shifts of zero. As already discussed, this is equivalent to performing subspace iteration on V_{r-i}. The use of implicit shifts of zero is also discussed by Meerbergen and Spence (1997) and will be reported on in our numerical experiments (Section 9).

3 Introduction to matrix transformations

Before we can apply an iterative eigensolver, we transform the generalized eigenvalue problem $Ax = \lambda Bx$ into a standard eigenvalue problem of the form

$$Tx = \theta x. \tag{3.1}$$

We need to do this because iterative methods such as subspace iteration or Arnoldi's method cannot be used directly to solve the generalized eigenvalue problem. It is well known that iterative eigensolvers rapidly provide approximations to well-separated extremal eigenvalues. When the eigenproblem arises from the spatial discretization of a partial differential equation, the sought-after eigenvalues (in our application, those of smallest real part) are generally not well separated. This results in slow convergence of the iterative method and, indeed, the method may never provide good approximations to the wanted eigenvalues. We therefore want to choose T to have the following properties:

- The sought-after eigenvalues of (A, B) should be transformed to well-separated extremal eigenvalues of T.
- The wanted left-most eigenvalue λ_1 of (A, B) must be easily recoverable from the dominant eigenvalue of T.
- For any v, $w = Tv$ should be efficiently computed.

We shall denote the eigenvalues of T by θ_i $(i = 1, \ldots, n + m)$ and we will assume that these eigenvalues are ordered by decreasing order of their absolute values i.e. $i > j \Rightarrow |\theta_i| \leq |\theta_j|$.

For the generalized eigenvalue problem, $Ax = \lambda Bx$, rational transformations are an obvious choice because the solution of some linear system involving A, B and/or a linear combination of A and B is needed. In this report, we study shift-invert and Cayley transformations.

4 Shift-invert techniques

In this section, we review the use of the shift-invert spectral transformation and the many associated details for the eigenvalue problem (1.2).

4.1 The shift-invert transformation

If we subtract σB ($\lambda \neq \sigma$) from both sides of $Ax = \lambda Bx$ and then postmultiplying by $(A - \sigma B)^{-1}(\lambda - \sigma)^{-1}$,

$$(A - \sigma B)^{-1} Bx = \theta x, \qquad \theta = (\lambda - \sigma)^{-1} \tag{4.1}$$

results. The matrix

$$T_{SI}(\sigma) = (A - \sigma B)^{-1} B \tag{4.2}$$

is termed the shift-invert transformation, and was first introduced by Ericsson and Ruhe (1980) for use in a Lanczos method. The scalar σ is referred to as the *shift* or *pole*. Since the eigenvectors of (A, B) and T_{SI} are identical, the relationship

$$\lambda = \sigma + \frac{1}{\theta} \tag{4.3}$$

can be used to recover the eigenvalues of (A, B) from those of the transformed problem. The shift-invert transformation combined with an iterative eigensolver can be used to find eigenvalues of (A, B) lying close to σ because eigenvalues close to σ are mapped away from the origin while those lying far from σ are mapped close to zero.

For the shift-invert transformation to be suitable for finding complex eigenvalues with arbitrary imaginary part, the shift σ must be complex. One possibility is to work entirely in complex arithmetic but, in our application, the matrices A and B are real. The eigenvalues of (A, B) come in complex conjugate pairs and so it is desirable to use algorithms which compute complex conjugate pairs. This is because, if there are pairs of eigenvalues lying close to one another, it may be difficult to match the pairs if the conjugates are only computed approximately. A wrong match can give incorrect eigenvectors. The alternative is to work entirely using real arithmetic. This is discussed by Parlett and Saad (1987). Maintaining real arithmetic can only be done at the cost of either affecting sparsity or doubling the dimension of the problem. Because of these disadvantages that result from using complex shifts, in this study we do not consider their use.

4.2 Shift-invert theory for the discretized Navier Stokes equations

In this section we consider the discretized linearized Navier Stokes equations (1.2–1.3). From a theoretical point of view, no generality is lost by taking the pole σ in the shift-invert transformation to be zero. In this case we define the operator S to be

$$S = T_{SI}(0) = A^{-1}B. \tag{4.4}$$

The following result is given by Meerbergen and Spence (1997) (see also Malkus, 1981 and Ericsson, 1986).

Theorem 1 *S defined by (4.4) has $n-m$ nonzero eigenvalues, a zero eigenvalue of algebraic multiplicity $2m$ and geometric multiplicity m. The order of the Jordan blocks corresponding to the defective eigenvalue 0 is two. The null space $\mathcal{N} \equiv Null(S) = Null(B)$ has dimension m and the generalized null space $\mathcal{G} \equiv Null(S^2)\backslash Null(S)$ also has dimension m. $\mathcal{N}$ and $\mathcal{G}$ satisfy $S\mathcal{G} = \mathcal{N}$ and $S^2\mathcal{G} = S\mathcal{N} = 0$, and, if $\mathcal{R} := Range(S^2)$, $\mathbf{C}$ can be represented as a direct sum of $\mathcal{N}$, $\mathcal{G}$, and $\mathcal{R}$.*

In exact arithmetic, when Arnoldi's method is applied with a starting vector $v_1 \in \mathcal{R}$ then only approximations to the nonzero eigenvalues of S can be computed. Such an initial vector can be chosen as $v_1 = S^2 v$ with v arbitrary because $S^2(\mathcal{N} + \mathcal{G}) = 0$. However, in practice, rounding errors introduce components in $\mathcal{N} + \mathcal{G}$ which corrupt the approximate eigenvalues and eigenvectors. The paper by Meerbergen and Spence (1997) looks at the efficient control of these unwanted directions and we discuss this in the rest of this section.

From (1.2–1.3), we see that S has the structure

$$S = \begin{pmatrix} S_1 & 0 \\ S_2 & 0 \end{pmatrix} \quad S_1 \in \mathbf{R}^{n\times n},\ S_2 \in \mathbf{R}^{m\times n}. \tag{4.5}$$

Therefore, if $x = \begin{pmatrix} u \\ p \end{pmatrix}$, $u \in \mathbf{C}^n$, $p \in \mathbf{C}^m$ is an eigenvector of S, then

$$S_1 u = \theta u \quad \text{and} \quad S_2 u = \theta p. \tag{4.6}$$

If $\theta \neq 0$ (λ is finite), then

$$p = \theta^{-1} S_2 u \tag{4.7}$$

and the reduced problem

$$S_1 u = \theta u \tag{4.8}$$

may be used to determine the nonzero eigenvalues and corresponding eigenvectors of S. (Since $S_1 \in \mathbf{R}^{n\times n}$, there still remains m zero eigenvalues.) We did not consider solving this reduced eigenvalue problem because it involves applying a projection that includes $(C^T K^{-1} C)^{-1}$.

The standard Arnoldi method uses the classical inner product $x^H y$. For the generalized eigenvalue problem $Ax = \lambda Bx$, if the matrix B is symmetric positive semi-definite, the B semi-inner product $x^H By$ may be used instead. The so-called B-orthogonal Lanczos method has been used by Ericsson (1986) and Nour-Omid, Parlett, Ericsson and Jensen (1987) for the symmetric generalized eigenvalue problem; Meerbergen and Spence (1997) extend its use to the nonsymmetric problem. They show that the B-orthogonal method applied to S is equivalent to the M-orthogonal method applied to S_1 and that the H_r produced by Arnoldi's method is only contaminated by $\mathcal{G}$ components in the Arnoldi vectors. Furthermore, their analysis also shows that the p component of the eigenvector x does not play a role in the B-orthogonal Arnoldi method and therefore cannot be guaranteed to be correct.

The p component of x may be computed by applying S to x. This can be achieved with an implicit application of S. Observe that if $x = V_r y$ with $H_r y = \theta y$ then a formal step of inverse iteration with S gives

$$Sx = V_r H_r y + f_r e_r^T y = x\theta + f_r e_r^T y. \tag{4.9}$$

Ericsson and Ruhe used the above implicit application of S to improve the quality of the eigenvector x. Following Meerbergen and Spence (see also Nour-Omid et al. (1987)), we use the expression *purification of* x to refer to the operation of applying S (or more generally, T_{SI}) to the vector x. The purified vector is $z = x + f_r e_r^T y/\theta$.

In summary, two applications of S are required to produce approximate eigenvectors and eigenvalues that are not corrupted by components in $\mathcal{N}$ or $\mathcal{G}$. In exact arithmetic, a starting vector in the range of S^2 avoids any possible corruption. However, as usual, rounding errors complicate the situation. Meerbergen and Spence propose an approach that is a combination of the B-orthogonal implicitly restarted Arnoldi method and purification. Recall from Section 2 that an implicit restart with a zero shift is equivalent to employing a starting vector Sv_1. Thus using a zero shift has the effect of removing the $\mathcal{N}$ component and maps the $\mathcal{G}$ component into $\mathcal{N}$. This remaining $\mathcal{N}$ component may be removed by a second implicit application of S using (4.9).

5 Cayley transformations

The use of generalized and modified Cayley transformations for the eigenvalue problem (1.2) are addressed.

5.1 Generalized Cayley transformation

Given real numbers σ and μ with $\sigma \neq \lambda_i$ $(i = 1, \ldots, n-m)$, the generalized Cayley transformation T_C (see, for example, Garratt, Moore and Spence, 1991) is defined by

$$T_C(\sigma, \mu) = (A - \sigma B)^{-1}(A - \mu B), \quad \sigma < \mu. \tag{5.1}$$

We term σ the *pole* and μ the *zero*. This is a generalization of the standard Cayley transformation where $\mu = -\sigma$ (Franklin, 1968) and a special case of the Möbius transformation

$$(cA + dB)^{-1}(aA + bB), \quad ad - bc \neq 0. \tag{5.2}$$

The relationship between the finite eigenvalues of (A, B) and those of the transformed problem is

$$\lambda = \frac{\sigma\theta - \mu}{\theta - 1}. \tag{5.3}$$

The usefulness of the generalized Cayley transformation lies in the fact that

$$\begin{aligned} \mathrm{Re}(\lambda) &< \frac{1}{2}(\sigma + \mu) \quad \Leftrightarrow |\theta| > 1 \\ \mathrm{Re}(\lambda) &\geq \frac{1}{2}(\sigma + \mu) \quad \Leftrightarrow |\theta| \leq 1 \end{aligned} \tag{5.4}$$

(see Garratt, 1991). Eigenvalues lying far from σ and μ are mapped close to $+1$. This includes eigenvalues with large real parts as well as those with large imaginary parts. In many applications arising from discretizations of partial differential equations, a significant proportion of the eigenvalues of (A, B) have a large positive real part and only a small number of eigenvalues lying close to the imaginary axis are of interest in stability analysis. Since the former map close to unity and (small) changes in the pole and zero have little effect on this, one way that σ and μ may be chosen is to place the first unwanted eigenvalue θ_{s+1} $(s \geq 1)$ on the unit circle and then maximize the distance of the dominant eigenvalue θ_1 from the unit circle. With this choice, Arnoldi's method can be expected to rapidly provide approximations to the sought-after eigenvalues.

The Cayley transformation (5.1) can also be written in the form

$$T_C(\sigma,\mu) = I + (\sigma - \mu)T_{SI}(\sigma), \tag{5.5}$$

where $T_{SI}(\sigma)$ is the shift-invert transformation given by (4.2). Thus the Cayley transformation is a scaled and translated shift-invert transformation. Since in exact arithmetic Arnoldi's method is translation invariant (see Parlett, 1980), the two formulations (5.1) and (4.2) are equivalent in the sense that $\mathrm{Range}(V_r(C)) = \mathrm{Range}(V_r(SI))$ (provided the poles of the transformations are equal and the same starting vectors are used).

The discussion at the end of section 4.2 explained how we can remove contamination by $\mathcal{N}$ and $\mathcal{G}$ when using a shift-invert transformation. Therefore, from (5.5) we deduce that an implicit restart with a shift of one removes the $\mathcal{N}$ and maps the $\mathcal{G}$ to a $\mathcal{N}$ one. To remove this final $\mathcal{N}$ component, an implicit application with $T_C - I$ is performed. This also ensures that the p component of the computed eigenvector is correct.

5.2 Modified Cayley transformation

A major drawback of the generalized Cayley transformation T_C is that the infinite eigenvalues of the discretized Navier Stokes equations are mapped to $+1$. It is anticipated that this may cause numerical difficulties because, in general, eigenvalues at $+1$ lie in the outer part of the spectrum of the transformed problem and, as a result, approximations to these eigenvalues are likely to be computed by iterative methods such as Arnoldi's method. To try and overcome this, Cliffe et al. (1993) propose using a modified Cayley transformation to solve (1.2). They define the modified problem

$$\begin{pmatrix} K - \mu M & \beta C \\ \beta C^T & 0 \end{pmatrix} \begin{pmatrix} u \\ q \end{pmatrix} = \theta \begin{pmatrix} K - \sigma M & C \\ C^T & 0 \end{pmatrix} \begin{pmatrix} u \\ q \end{pmatrix}, \tag{5.6}$$

with β a real scalar. Clearly, $\beta = 1$ corresponds to the generalized Cayley transformation applied to the discretized Navier Stokes equations. The modified Cayley transformation T_M is defined by

$$T_M(\sigma,\mu,\beta) = (A(1) - \sigma B)^{-1}(A(\beta) - \mu B). \tag{5.7}$$

where $A(\beta)$ denotes the matrix on the left-hand side of (5.6).

It is straightforward to prove the following result for the eigenvalues of the modified problem (5.6) (see Garratt, 1991).

Theorem 2 *Assume $\sigma \neq \lambda_i$ $(i = 1, \ldots, n-m)$. If λ is an eigenvalue of the discretized Navier Stokes equations (1.2), then $(\lambda - \mu)/(\lambda - \sigma)$ is an eigenvalue of the modified problem (5.6). In addition, (5.6) has m eigenvalues β, each with algebraic multiplicity* 2 *and geometric multiplicity* 1.

The parameter β is chosen so that the infinite eigenvalues are mapped inside the unit circle, where they are much less likely to be computed by an iterative eigensolver. An obvious choice is $\beta = 0$ and this is the value which will be used in our numerical experiments.

Garratt also gives the following simple relationship between the eigenvectors of the original problem (1.2) and those of the modified problem (5.6).

Theorem 3 *Assume $\sigma \neq \lambda_i$ and $\beta \neq (\lambda_i - \mu)/(\lambda_i - \sigma)$ $(i = 1, \ldots, n-m)$, then $(\lambda, \begin{pmatrix} u \\ p \end{pmatrix})$ is an eigensolution of the original problem (1.2) if and only if $(\theta, \begin{pmatrix} u \\ q \end{pmatrix})$ is an eigensolution of the modified problem (5.6) where*

$$\theta = (\lambda - \mu)/(\lambda - \sigma) \notin \{1, \beta\}, \quad q = (\theta - 1)/(\theta - \beta)p. \tag{5.8}$$

Using this theorem, approximate eigenvectors of (1.2) can easily be obtained from the computed eigenvectors of (5.6).

In practice, regardless of the Cayley transformation employed, the Cayley parameters σ and μ are updated at each restart. This allows rapid computation of the left-most eigenvalues. If we denote by $\sigma(j)$ and $\mu(j)$ the Cayley parameters during restart or iteration j, then an important consequence of the above result is that the eigenvectors computed with $\sigma = \sigma(j)$ and $\mu = \mu(j)$ are *not* identical to those with $\sigma = \sigma(j+1)$ and $\mu = \mu(j+1)$ (unless the poles and shifts remain the same). Using Theorem 3 it can be shown that the eigenvectors of (5.6) with $\sigma(j+1)$ and $\mu(j+1)$ can be obtained from those with $\sigma(j)$ and $\mu(j)$ by an appropriate scaling of the q term of the eigenvectors. Full details are given by Garratt (1991) (page 98).

We would like to avoid having to scale the eigenvectors at the start of each iteration. We now discuss how this can be done. With the choice $\beta = 0$, the modified problem (5.6) can be rewritten in the form

$$\hat{S} \begin{pmatrix} u \\ q \end{pmatrix} = \begin{pmatrix} \hat{S}_1 & 0 \\ \hat{S}_2 & 0 \end{pmatrix} \begin{pmatrix} u \\ q \end{pmatrix} = \theta \begin{pmatrix} u \\ q \end{pmatrix}, \tag{5.9}$$

where $\hat{S}$ has the same structure as the shift-invert operator S for the discretized Navier Stokes equations described in Section 4.2. We also note that when using the modified Cayley transformation in the B-orthogonal Arnoldi method, the q component of the eigenvector does not play a role and the correct eigenvector is obtained by post-processing with $\hat{S}$. The required eigenvector $\begin{pmatrix} u \\ p \end{pmatrix}$ of the discretized Navier Stokes equations can then be computed using Theorem 3. However, as it is p and not q that is required, we may compute p directly by purifying with T_{SI}.

More generally, using either the standard inner product or the B semi-inner product, we can obtain the eigenvector of the original problem by applying T_{SI} to the computed eigenvector of the modified problem.

6 Cayley transform Arnoldi

In this section, we summarize the algorithm we use for the solution of the discretized linearized Navier Stokes equations. We combine shift-invert and Cayley transformations with Arnoldi's method. Shift-invert is used to get an initial approximation to the spectrum and is also used to purify the computed eigenvectors. Various implementation details are discussed. In Sections 7 and 8 we look at the linear equation solver and the use of the ARPACK software to implement our algorithm.

6.1 Algorithm outline

We first present an outline of our Cayley transform Arnoldi algorithm. Here we assume that the number of sought-after eigenvalues is s and the number of Arnoldi vectors to be generated at each iteration is r.

In an abuse of notation, we denote the approximate eigenvalues and eigenvectors computed for T_{SI} or T_C by θ and x, respectively. Hence, λ is a θ that is mapped back to an approximate eigenvalue of the original eigenvalue problem (1.2).

Shift-invert iteration:

1. Factorize $A = LU$
2. Choose v_1 randomly and normalize.
3. Compute $v_1 \Leftarrow S^2 v_1$ $(S = A^{-1}B)$ and normalize.

```
 4. Compute $\theta_1, \theta_2, \ldots, \theta_r$ by computing an Arnoldi reduction of length $r$
    for $S$.
 5. Let $\lambda_i = 1\backslash\theta_i$, $i = 1, 2, \ldots, r$.
 6. Order $\lambda_i$ in increasing order of their real parts.

Cayley iterations:
 7. Choose $v_1$ randomly and normalize.
 8. Compute $v_1 \Leftarrow S^2 v_1$ and normalize.
 9. repeat until convergence
     (a) Choose $\sigma < \mu$ with $(\sigma + \mu)/2 = Re(\lambda_{s+1})$.
     (b) Factorize $A - \sigma B = LU$.
     (c) Compute $\theta_1, \theta_2, \ldots, \theta_r$ by computing an Arnoldi reduction of length
         $r$ for $T_C$.
     (d) Let $\lambda_i = T_C^{-1}(\theta_i)$, $i = 1, 2, \ldots, r$.
     (e) Order $\lambda_i$ in increasing order of their real parts.
     (f) Construct a new starting vector $v_1$ by implicitly restarting the
         Arnoldi reduction of length $r$.
10. Compute eigenvectors $x_i$ of $T_C$ corresponding to the converged eigen-
    values.
11. Obtain eigenvectors of $(A, B)$ by purifying $x_i \Leftarrow T_{SI}(\sigma) x_i$.
```

We observe that shift-invert with a zero pole is used to compute an initial approximation to the spectrum of (A, B). Zero is an appropriate choice for the pole because the interest is in the eigenvalues lying close to the imaginary axis. We do **not** test for convergence after the shift-invert step because the eigenvalues close to the origin may not be the left-most eigenvalues. We thus always force at least one step of the Cayley iteration to be performed. During the Cayley iterations, we check at each iteration that σ has changed since the previous iteration to avoid refactorizing $A - \sigma B$ unnecessarily.

In our numerical experiments, we used the generalized and modified Cayley transformations in conjunction with the standard inner or B semi-inner products.

6.2 Convergence testing

We have to decide when the computed eigenvalues and eigenvectors are good approximations to those of (A, B). We proceed to determine this in two steps. The first step checks if θ and x are acceptable as approximations to an eigenpair of T_C. The second step checks that θ and x produce an acceptable approximation to an eigenpair of (A, B).

Let $x = V_r y$ with $H_r y = \theta y$, where $\|y\|_2 = 1$. Although the direct residual $\|T_C x - \theta x\|$ could be computed, this would involve additional applications of T_C. However,

$$T_C x - \theta x = T_C V_r y - V_r H_r y = f_r \, e_r^T y, \tag{6.1}$$

and so the Ritz estimate $\|f_r\| \, |e_r^T y|$ is equal to the direct residual. ARPACK accepts θ as a good approximation if $\|f_r\| \, |e_r^T y| \leq |\theta| \epsilon_U$, where ϵ_U is a user-specified tolerance. We remark that ARPACK uses $\|f_r\|_B$ or $\|f_r\|_2$ depending on the inner product used. Since the eigenvalues of interest are the eigenvalues of T_C of largest modulus, normalizing the residual by $|\theta|$ takes into account the scale of the data.

We now discuss how to check whether θ and x provide a good approximation to an eigenpair of (A, B). A simple rearrangement of (6.1) results in

$$Ax - \frac{\sigma\theta - \mu}{\theta - 1} Bx = -\frac{e_r^T y}{\theta - 1}(A - \sigma B) f_r, \tag{6.2}$$

and this is the residual of the computed eigenpair in the original system. However, as discussed in Section 5.1, purification of x is performed—to remove nullspace components of B and as an inexpensive means of improving the quality of x—via an implicit application of T_{SI} or, equivalently, via $T_C - I$. Using (6.1),

$$(T_C(\sigma, \mu) - I)x = (\theta - 1)x + e_r^T y f_r. \tag{6.3}$$

The approximate eigenvector x is replaced by the purified vector $z = x + e_r^T y/(\theta - 1) f_r$. From (6.2), it is straightforward to show that the residual of the purified vector in the original system is

$$Az - \frac{\sigma\theta - \mu}{\theta - 1} Bz = \frac{\mu - \sigma}{(\theta - 1)^2}(e_r^T y) B f_r. \tag{6.4}$$

Note that $\|z\|^2 = 1 + \frac{|e_r^T y|^2}{|\theta - 1|^2} \|f_r\|^2$.

From Equation (6.1), we see that x is orthogonal to the residual $T_C x - \theta x$. This Galerkin condition is lost upon transforming the computed eigenpair to the original system—whether we use x or z. This is because $\frac{\sigma\theta-\mu}{\theta-1}$ is not the Rayleigh quotient associated with x or z. If the B-orthogonal Arnoldi method is used, then from (6.4)

$$|\frac{z^H A z}{z^H B z} - \frac{\sigma\theta - \mu}{\theta - 1}| = (\mu - \sigma)\frac{|e_r^T y|^2}{|\theta - 1|^3}\frac{\|f_r\|_B^2}{\|z\|_B^2} \le (\mu - \sigma)\frac{|e_r^T y|^2}{|\theta - 1|^3}\|f_r\|_B^2. \quad (6.5)$$

The last inequality follows because $\|z\|_B^2 \ge 1$. In other words, the approximate eigenvalue $(\sigma\theta - \mu)/(\theta - 1)$ is nearly a Rayleigh quotient for (A, B) when the purified vector z is used as the approximate eigenvector. On the other hand, from Equation (6.2) we deduce that

$$|x^H A x - \frac{\sigma\theta - \mu}{\theta - 1}| = \frac{|e_r^T y|}{|\theta - 1|} x^H A f_r \quad (6.6)$$

is the error in using $(\sigma\theta - \mu)/(\theta - 1)$ as a Rayleigh quotient for (A, B) when the unpurified eigenvector x is used.

Equations (6.4) and (6.5) imply that the purified vector z is a better approximation than x for the eigenvector associated with the approximate eigenvalue $(\sigma\theta - \mu)/(\theta - 1)$ provided that $|\theta - 1|$ is greater than one. Moreover, when $|\theta - 1| > 1$, only a moderately-sized Ritz estimate is needed to achieve a small direct residual, a small Rayleigh quotient error, and $\|z\| = 1$ up to second order terms. If the pole σ is near the left-most eigenvalues, then these left-most eigenvalues are mapped by T_C to large eigenvalues and hence $|\theta - 1| > 1$.

Our numerical experiments measure the direct residual using both x and z with $\lambda = (\sigma\theta - \mu)/(\theta - 1)$. We define the *relative residual* of an approximate eigenvector u to be

$$\|Au - \lambda B u\|_2 / \|u\|_2. \quad (6.7)$$

We employ the Euclidean norm for the relative residual regardless of whether `ARPACK` is used with the standard inner or B semi-inner product so that the pressure component of the approximate eigenvector can be checked. Section 8.1 will give details of how `ARPACK` normalizes the computed eigenvectors.

6.3 Missing eigenvalues

It is necessary to exercise some caution to avoid accepting an approximation to an eigenvalue that is not the one of smallest real part. Recall that the parameters σ and μ are chosen using the latest approximations to the eigenvalues. Suppose the left-most eigenvalue λ_1 is complex with a large imaginary part and λ_3 is real with $\lambda_3 \approx \mathrm{Re}(\lambda_1)$, and let θ_i be the eigenvalue of the transformed problem corresponding to λ_i. We have observed during numerical experiments that $|\theta_3| > |\theta_1|$ with θ_1 close to $+1$. In this case, Arnoldi's method rapidly produces an accurate θ_3 and, without further checks, the corresponding eigenvalue λ_3 is accepted as the left-most eigenvalue of the original problem. We stress that this is a result of only having available approximations to the eigenvalues when selecting σ and μ. This problem was also observed by Garratt (1991) and Meerbergen and Roose (1996).

In an attempt to overcome the problem of accepting the wrong eigenvalue as the left-most one, we introduce an additional test on the computed eigenvalues. Once the dominant eigenvalues (largest in magnitude) of the transformed problem are acceptable approximations, we order the r eigenvalues of the transformed problem in decreasing order of their moduli. We then compute the corresponding r eigenvalues of the original problem and order them in increasing order of their real parts. By comparing the two orderings we are able to check whether or not any computed eigenvalues lie to the left of those that are acceptable approximations. If there any such eigenvalues, we increase the number s of eigenvalues requested. Although this strategy is not guaranteed to find eigenvalues that have been missed, it can help avoid accepting the wrong eigenvalue. Numerical experiments in Section 9 demonstrate that our strategy is effective.

6.4 Spurious eigenvalues

Recall from Theorem 2 with $\beta = 1$ that the generalized Cayley transformation applied to the discretized Navier Stokes equations has $2m$ eigenvalues at $+1$ corresponding to the infinite eigenvalues of (1.2). These eigenvalues are not relevant for the stability analysis but are likely to be computed because they lie in the outer part of the spectrum of the transformed problem. It is important when updating the Cayley parameters σ and μ that we do not use these spurious eigenvalues. The eigenvalues used in choosing σ and μ are the current $s+1$ left-most eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{s+1}$. From (5.3), it follows that if $\theta = 1 + \delta$ with $\delta > 0$ small, λ will be large and negative and will be used in selecting the Cayley parameters. On each iteration we therefore exclude all real θ_i which lie close to 1. Our numerical experiments showed that if we did not do this, the IRAM did not produce approximations to the wanted eigenvalues. We also exclude spurious eigenvalues when we search for possible missing eigenvalues.

7 The linear equation solver

The efficiency of an iterative eigensolver for the generalized eigenvalue problem depends on the efficiency of the method used to solve linear systems of the form

$$(A - \sigma B)x = b. \tag{7.1}$$

Either a direct method or an iterative method may be used. For very large problems, direct methods can be prohibitively expensive in terms of both time and memory requirements. However, there are also difficulties associated with selecting and using iterative methods for eigenvalue computations and they have not yet been widely adopted for the solution of industrial problems (see Meerbergen and Roose, 1996 for a discussion and references). In this study, we use a frontal method for the solution of (7.1). One important reason for this choice is that, in the applications of interest to us, finite-element discretizations are used and the matrices A, B are available as unassembled finite element matrices of the form

$$A = \sum_{l=1}^{nelt} A^{(l)}, \quad B = \sum_{l=1}^{nelt} B^{(l)}, \tag{7.2}$$

where $A^{(l)}$ and $B^{(l)}$ are nonzero only in those rows and columns that correspond to variables in the lth element. The recent study by Duff and Scott (1996*a*) has shown that for efficiency in terms of the factorization time and sparsity of the matrix factors, it is important not to assemble the matrices prior to solving the linear system but the element form should be exploited. Using the element form also allows us to solve much larger systems than might otherwise be possible because each element matrix $A^{(l)}$ and $B^{(l)}$ need only be generated as it is required, substantially reducing storage requirements. Storage requirements can be further reduced by holding the matrix factors out-of-core. The only code for nonsymmetric systems we currently have available in the Harwell Subroutine Library (Harwell Subroutine Library 1996) that allows both element input and (optional) out-of-core storage is the frontal solver `MA42` of Duff and Scott (1993, 1996*b*), and this is the code we use in our numerical experiments. For simplicity of notation, in the following brief discussion of frontal schemes, we assume we are solving $Ax = b$ (that is, $\sigma = 0$).

The frontal method (Irons 1970, Hood 1976, Duff 1984) is a variant of Gaussian elimination and involves the matrix factorization

$$A = PLUQ,$$

where P and Q are permutation matrices, and L and U are lower and upper triangular matrices, respectively. The solution process is completed by performing the forward elimination

$$PLy = b, \tag{7.3}$$

followed by the back substitution

$$UQx = y. \tag{7.4}$$

The method, although originally developed by Irons for symmetric positive definite systems, can be used for symmetric and nonsymmetric systems. If a_{ij} and $a_{ij}^{(l)}$ denote the (i,j)th entry of A and $A^{(l)}$, respectively, the basic assembly operation when forming A is of the form

$$a_{ij} \Leftarrow a_{ij} + a_{ij}^{(l)}. \tag{7.5}$$

It is evident that the basic operation in Gaussian elimination

$$a_{ij} \Leftarrow a_{ij} - a_{iq}[a_{qq}]^{-1}a_{qj} \tag{7.6}$$

may be performed as soon as all the terms in the triple product (7.6) are *fully summed* (that is, are involved in no more sums of the form (7.5)). The frontal method interleaves the assembly and Gaussian elimination processes and avoids the explicit assembly of the matrix A. This allows all intermediate computation to be performed on a dense matrix, termed the *frontal matrix*, whose rows and columns correspond to variables that have not yet been eliminated but occur in at least one of the elements that have been assembled. By working within the frontal matrix, it is possible to use dense linear algebra kernels and, in particular, the Level 3 BLAS can be exploited (Dongarra, DuCroz, Duff and Hammarling 1990). The use of BLAS in `MA42` is described in Duff and Scott (1996*b*).

In practice, for general systems of equations, stability considerations may delay some eliminations. `MA42` uses a threshold criterion of the from

$$|a_{lk}| \geq u \cdot max_i|a_{ik}| \tag{7.7}$$

where the stability threshold $u \in (0,1]$ is a parameter chosen by the user. Note that using a small value of u will lead to few delays and minimize the number of entries in the factors while a larger value of u (for example, the default value $u = 0.1$) improves stability. The choice $u = 1.0$ corresponds to partial pivoting.

By holding the matrix factors in direct access files, the frontal method can solve quite large problems with modest amounts of high-speed memory. We remark that, because the size of the frontal matrix increases when a variable appears for the first time and decreases whenever it is eliminated, the order where the elements are assembled has a crucial effect on the storage requirements and on the number of floating-point operations. Elements should be preordered to reduce the size of the frontal matrices. In our experiments the elements are preordered using the Harwell Subroutine Library code `MC43`.

Once the matrix factors have been formed and stored, the `MA42` package has a separate entry, `MA42C`, that uses the factors for solving a linear system with multiple right-hand sides b. As well as using high level BLAS in the factorization, the BLAS are used when performing the forward elimination (7.3) and the backward substitution (7.4). When solving for a single right-hand side, the Level 2 BLAS are used but if there are multiple right-hand sides, the Level 3 BLAS are used. Since greater efficiency is achieved by using the Level 3 BLAS and the factors have only to be read in once for each call to `MA42C`, the performance of `MA42` improves with the number of right-hand sides. This is illustrated in Section 9 (see also Duff and Scott, 1993).

When using a shift-invert or Cayley transformation, the matrix $A - \sigma B$ must be refactorized each time the pole σ is updated. A disadvantage of using Arnoldi's method as the eigensolver is that it requires the repeated solution of linear systems with a single right-hand side. This is in contrast to the subspace iteration method, where the number of right-hand sides is equal to the subspace dimension r. Partly because of this, Lehoucq and Maschhoff (1997) have recently developed a block version of the implicitly restarted Arnoldi method.

8 The software package ARPACK

In this section, we summarize `ARPACK` and the modifications we made so that Cayley transformations could be used.

8.1 Introduction to ARPACK

The `ARPACK` software package (Lehoucq et al., 1998) provides subroutines that implement the implicitly restarted Arnoldi metho (IRAM). `ARPACK` was developed for finding a few eigenvalues of large-scale symmetric, nonsymmetric, standard or generalized eigenvalue problems (complex arithmetic versions are available). An important feature of the package is the reverse communication interface. This feature provides a convenient way to interface with application codes without imposing a structure on the user's matrix or on the way in which matrix-vector products are computed. In particular, if the matrix is not available explicitly, the user is free to express the action of the matrix on a vector through a subroutine call or code segment. This makes the code attractive for our applications where the matrices A and B are unassembled finite-element matrices (see (7.2)). For large problems, there may be insufficient storage to hold all the element matrices $A^{(l)}$ and $B^{(l)}$ in-core. The use of an eigensolver which does not require the matrices to be held using a prescribed format is therefore essential.

Another important feature of `ARPACK` is that full numerical orthogonality (to machine precision) of the Arnoldi basis vectors is maintained. A point we wish to emphasize is that the cost of this orthogonality often represents less than 5 % of the total cost of the eigensolver. For large-scale problems, the dominant cost is that of performing matrix-vector products.

There are many different options included within the `ARPACK` package. Here we briefly mention those that are useful for solving the problems of interest to us: full details of `ARPACK` are given in Lehoucq et al. (1998).

- The user may select the implicit shifts in the implicitly shifted QR algorithm performed during each iteration. This allows implicit shifts of zero to be used.
- The initial starting vector v_1 may be chosen by the user. In our experiments, we want to purify the starting vector before using `ARPACK` and this option enables us to do this.
- The default convergence tolerance ϵ_U used by `ARPACK` is machine precision. Since our main interest is in determining whether the left-most eigenvalue has a positive or negative real part, we do not need to compute the eigenvalues to a large number of decimal places. Therefore, in our experiments we will use a larger convergence tolerance (see Section 9).

- When using `ARPACK` to solve the generalized eigenvalue problem, $Ax = \lambda Bx$, where B is a symmetric positive semi-definite matrix, the B semi-inner product may be used. We will use both the standard inner product and the B semi-inner product.
- For the generalized eigenvalue problem, the user has the option of using a shift-invert transformation $T_{SI}(\sigma)$ or, if A is symmetric, a standard Cayley transformation $T_C(\sigma, -\sigma)$. If the shift-invert or Cayley mode is used, `ARPACK` accepts a computed eigenvalue and eigenvector of T_{SI} or T_C if the associated Ritz estimate (6.1) is sufficiently small. See the discussion after (6.1).
- Eigenvectors may be computed on request once approximations to the sought-after eigenvalues have converged. If eigenvectors are requested and a shift-invert or Cayley transformation has been employed, the computed eigenvalues are mapped to those of the original system.

We conclude by explaining how `ARPACK` normalizes the computed eigenvectors. If the standard inner product is used, the computed eigenvector u corresponding to a real computed eigenvalue is normalized so that $\|u\|_2 = 1$; if the B semi-inner product is used, then $\|u\|_B = 1$. When the eigenvectors corresponding to a computed complex conjugate pair of eigenvalues are computed, the real and imaginary parts, u_R and u_I, of the vector associated with the computed eigenvalue with positive imaginary part are stored. If the standard inner product is used, u is normalized so that $u_R^T u_R + u_I^T u_I = 1$; if the B semi-inner product is used, then $u_R^T B u_R + u_I^T B u_I = 1$.

8.2 Modifications to ARPACK

To use `ARPACK` for computing the left-most eigenvalues of the discretized Navier Stokes problems, it was necessary to make a small number of modifications to the package. These modifications essentially involved making the existing reverse communication interface more flexible. This flexibility was necessary to accommodate our use of spectral transformations. As mentioned above, for the generalized eigenvalue problem $Ax = \lambda Bx$, the user may optionally use a shift-invert transformation $T_{SI}(\sigma)$. When using this option, it is assumed that the user has chosen the pole σ and that the *same* pole is used throughout the computation: a facility for updating the shift is not explicitly offered. There is also no option for using a Cayley transform when A is nonsymmetric. We had to make minor changes to `ARPACK` so that we could use the generalized or the modified Cayley transformation. In addition, we needed changes to enable us to switch between using shift-invert and Cayley, and to allow us to update the Cayley parameters at each iteration. Our modified codes are available by anonymous ftp from `ftp.caam.rice.edu` in the directory `pub/software/ARPACK/CONTRIBUTED`.

We observe that `MA42` does not assume the degrees of freedom are numbered contiguously from 1 to $n+m$. `MA42` is designed in this way because in many finite-element applications (including the examples used in our numerical experiments), the boundary conditions imply that some of the degrees of freedom are known and so do not appear in the element data passed to the linear solver. As a result, the largest integer used to index a variable is greater than the order of the system. However, `ARPACK` does require contiguous numbering and we therefore have to map between the global freedom numbers used by `MA42` and the local freedom numbers used by `ARPACK`. Using `MA42` with `ARPACK` also means that we have two reverse communication interfaces to deal with, and this adds to the programming complexity.

9 Numerical experiments

In this section, we present numerical results for three tests problems. The test problems were supplied to us by Simon Tavener of Pennsylvania State University and were obtained using the finite-element package ENTWIFE (Cliffe, 1996). ENTWIFE was developed by AEA Technology to solve discretized elliptic and parabolic partial differential equations using finite-element methods. The code is used to compute singular points such as limit points, symmetry-breaking bifurcation points, and Hopf bifurcation points of the steady solution set. The current interest is in stability of laminar flows both in expanding channels and pipes and past bodies in pipes and channels. ENTWIFE uses subspace iteration as its eigensolver and employs the frontal code `MA42` as its linear equation solver. Experience has shown the subspace iteration solver to be reliable but the method is too slow to solve the large problems (up to 250,000 degrees of freedom) that are now of interest. One of the major aims of this project was to investigate the reliability of implicitly restarted Arnoldi with a view to using ARPACK as the eigensolver in ENTWIFE.

The problems used in our tests, while not as large as those AEA Technology would like to solve, are typical of the problems of interest. All the tests were performed on a SUN Ultra 1 workstation using double precision arithmetic. All timings are CPU times in seconds. As explained at the end of section 6.2, the Euclidean norm is used for computing all residuals. In each test, we were seeking the two left-most eigenvalues.

In our tables of results, the residual is given both before and after purification, that is, before and after the computed eigenvectors $x = \begin{pmatrix} u \\ p \end{pmatrix}$ are premultipled by the shift-invert operator T_{SI}. Recall that for the modified Cayley transformation T_M, the p component of x is not guaranteed correct until premultiplication by T_{SI} has been performed and so, in this case, the residual is only given after purification.

We also point out that, as mentioned, `ARPACK` maintains full (numerical) orthogonality of all r Arnoldi vectors. It should be emphasized that the dominant costs in time when computing eigenvalues of our test problems are those associated with performing matrix factorizations, solving linear systems using the matrix factors, and performing matrix-vector products with $A(\beta)$ and B. Our experience was that the time required to maintain the full orthogonality of the Arnoldi vectors along with all the other costs associated with the `ARPACK` implementation of the IRAM represented only 2%–3% of the total computation time. This is why in our tables of results we only give the total time together with the times for the matrix factorizations, solving linear systems, and performing matrix-vector products.

9.1 Problem 1

The first problem we look at is that of two-dimensional double-diffusive convection in a box (see Ortega, 1973, chapter 8, and Garratt, 1991, page 102). The model used has the following non-dimensional parameters: the Prandtl number Pr, the Rayleigh number Ra, the salinity Rayleigh number Rs and τ, the ratio of solutal and temperature diffusivities. A mixed finite-element approximation is used, with nine-noded quadrilateral elements with biquadratic interpolation for velocities, temperatures, and salinities, and discontinuous piecewise linear interpolation for pressures. This leads to a system of equations of the form (1.1), where $u \in \mathbf{R}^n$ represents velocity, temperature, and salinity, and $p \in \mathbf{R}^m$ represents pressure. Using a 16×16 grid, there are a total of 4859 degrees of freedom. Although it is possible to solve the linearized stability problem analytically, the problems which are obtained by varying the parameters are ideally suited for testing the effectiveness of our eigensolver for detecting Hopf bifurcations. We compute the left-most eigenvalues using the parameter values $Rs = 2000$, $Pr = 10$, and $\tau = 10^{-2}$ and three different values of the Rayleigh number Ra. For each value, the left-most eigenvalues are listed below (see Cliffe et al., 1993).

- $Ra = 2440$:
  $\lambda_1 = 9.8696 \times 10^{-2}$, $\lambda_2 = 3.9478 \times 10^{-1}$,
  $\lambda_{3,4} = 3.9478 \times 10^{-1} \pm i2.4561 \times 10$.
- $Ra = 2480$:
  $\lambda_{1,2} = 4.7486 \times 10^{-2} \pm i2.4502 \times 10$, $\lambda_3 = 9.8696 \times 10^{-2}$.
- $Ra = 2520$:
  $\lambda_{1,2} = -3.5071 \times 10^{-1} \pm i2.4437 \times 10$, $\lambda_3 = 9.8696 \times 10^{-2}$.

The interest lies in the loss of stability as Ra increases. The values $Ra =$ 2440 and 2480 correspond to stable steady state solutions and $Ra = 2520$ is an unstable steady state. The change in stability is due to a complex pair of eigenvalues crossing the imaginary axis at a Hopf bifurcation at $Ra \approx$ 2484. We anticipate that $Ra = 2480$ may cause our eigensolver difficulties because $|\mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_3)|$ is small relative to $|\mathrm{Im}(\lambda_1)|$. For $Ra = 2440$ the left-most eigenvalues are real and shift-invert with a zero shift will be successful. However, for $Ra = 2480$ and 2520, shift-invert misses the left-most eigenvalues.

We also consider varying Rs, with $Ra = 2440$ and the remaining parameters unchanged.

- $Rs = 1900$:
  $\lambda_{1,2} = -4.6001 \times 10^{-1} \pm i3.3800 \times 10$, $\lambda_3 = 9.8696 \times 10^{-2}$.
- $Rs = 1950$:
  $\lambda_{1,2} = -7.5082 \times 10^{-3} \pm i2.4185 \times 10$, $\lambda_3 = 9.8696 \times 10^{-2}$.
- $Rs = 1975$:
  $\lambda_1 = 9.8696 \times 10^{-2}$, $\lambda_{2,3} = 2.2047 \times 10^{-1} \pm i2.4374 \times 10$,
  $\lambda_4 = 3.9478 \times 10^{-1}$.

$Rs = 1900$ and 1950 correspond to unstable steady state solutions while $Rs = 1975$ is a stable steady state. We expect that finding λ_2 will be difficult in the case $Rs = 1975$ since λ_2 has a large imaginary part and lies between 2 real eigenvalues with $|\mathrm{Re}(\lambda_1) - \mathrm{Re}(\lambda_2)|$ and $|\mathrm{Re}(\lambda_4) - \mathrm{Re}(\lambda_2)|$ small compared to $|\mathrm{Re}(\lambda_2)|$.

In Table 1 we present statistics for this problem for factorizing A and solving systems $Ax = b$ using `MA42`. We give results for the first set of parameter values given above ($Rs = 2000$, $Pr = 10$, $\tau = 10^{-2}$, $Ra = 2440$)

but similar results are obtained for the other parameter values used, Here and in other tables where the number of floating-point operations ("flops") are quoted, we count all operations (+,-,*,/) equally. Following advice from AEA Technology, here and elsewhere the threshold parameter u (7.7) is set to 10^{-7}. For our test examples, we found that this choice of u did not lead to any instabilities and gave significantly sparser factors than those obtained using the default value of 0.1. Sparse factors are important for the factorization and solve times.

Table 1: MA42 statistics for problem 1 ($Rs = 2000$, $Pr = 10$, $\tau = 10^{-2}$, $Ra = 2440$).

	$u = 10^{-7}$	$u = 10^{-1}$
Flops for factorization	1.19×10^8	1.17×10^9
Real factor storage (Kwords)	965	3037
Integer factor storage (Kwords)	92	217
Minimum in-core storage (Kwords)	23	286
Factorize time (secs)	6.7	113.8
Solve time: 1 right-hand side (secs)	0.5	1.5
Solve time: 10 right-hand sides (secs)	1.5	4.4

In Tables 2 and 3 we present results for problem 1, using the generalized Cayley transformation T_C and the modified Cayley transformation T_M. Since we want to locate on which side of the imaginary axis the left-most eigenvalues lie, we do not need to compute the eigenvalues to a large number of decimal places. However, if we do not choose the convergence tolerance small enough, wanted eigenvalues may be missed. We set the convergence tolerance in ARPACK to $\epsilon_U = 10^{-6}$ and the number r of Arnoldi vectors is set to 20. We see from the tables that Arnoldi's method rapidly produces good approximations and, except for the case $Rs = 1975$, only two LU factorizations are required, corresponding to the factorization of S for the shift-invert step and a single Cayley iteration. For $Rs = 1975$, the test described in Section 6.3 found that the left-most eigenvalues had been missed. At this point r was increased to 25 and the number of requested eigenvalues was increased to 4. The computation then continued and the correct eigenvalues converged. We remark that this check for missing eigenvalues and the subsequent increase in r and the number of sought-after eigenvalues is performed automatically within our code: no action is required by the user.

Table 2: Basic operations, times, and residuals for T_C and T_M for problem 1 with a range of values of Ra. Zero shifts and the B semi-inner product are used. For T_C, $\beta = 1.0$ and for T_M, $\beta = 0.0$.

	$Ra = 2440$		$Ra = 2480$		$Ra = 2520$	
	T_C	T_M	T_C	T_M	T_C	T_M
Basic operations:						
Factorizations	2	2	2	2	2	2
Linear solves	46	46	46	46	46	46
$A(\beta) * x$	20	20	20	20	20	20
$B * x$	126	126	127	127	127	127
$(A(\beta) - \mu B) * x$	1	1	1	1	1	1
Times (secs):						
Factorizations	14.6	14.2	14.3	14.1	14.2	14.0
Linear solves	23.9	24.9	24.0	24.2	24.6	25.1
$A(\beta) * x$	4.3	12.8	4.4	13.0	4.3	12.9
$B * x$	30.5	30.3	30.9	30.2	30.4	30.8
$(A(\beta) - \mu B) * x$	0.4	0.8	0.4	0.6	0.4	0.7
Total	75.4	84.8	75.7	83.7	75.5	85.1
Relative residuals:						
Before purification	1.74d-17		1.19d-08		1.17d-08	
After purification	1.61d-17	1.63d-17	5.90d-10	5.90d-10	4.65d-10	4.65d-12

Table 3: Basic operations, times, and residuals for T_C and T_M for problem 1 with a range of values of Rs. Implicit shifts of zero and the B semi-inner product are used. For T_C, $\beta = 1.0$ and for T_M, $\beta = 0.0$.

	$Rs = 1900$		$Rs = 1950$		$Rs = 1975$	
	T_C	T_M	T_C	T_M	T_C	T_M
Basic operations:						
Factorizations	2	2	2	2	5	5
Linear solves	46	46	46	46	119	119
$A(\beta) * x$	20	20	20	20	90	90
$B * x$	126	126	127	127	339	339
$(A(\beta) - \mu B) * x$	1	1	1	1	4	4
Times (secs):						
Factorizations	14.2	14.0	14.3	14.0	35.3	34.9
Linear solves	25.1	25.0	24.9	24.9	64.5	64.3
$A(\beta) * x$	4.3	12.8	4.3	12.8	19.2	58.0
$B * x$	30.0	30.2	30.1	30.2	80.1	81.4
$(A(\beta) - \mu B) * x$	0.4	0.7	0.4	0.8	1.7	2.6
Total	75.7	84.3	75.7	84.4	205.3	245.6
Relative residuals:						
Before purification	6.39d-09		8.74d-09		1.49d-08	
After purification	4.31d-10	4.31d-10	5.69d-10	5.69d-10	7.71d-10	1.49d-10

Our results show that, for problem 1, that when using implicit shifts of zero during the restart and the B semi-inner product, there is little to choose between the generalized and modified Cayley transformations. Both use the same number of factorizations and linear solves. However, in terms of time, T_M is slightly more expensive. This is accounted for by the way we hold the element data. For each of our test problems, the data supplied to us is in the form of the element matrices for

$$\begin{pmatrix} K & C \\ C^T & 0 \end{pmatrix}, \quad \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} M & C \\ C^T & 0 \end{pmatrix}. \tag{9.1}$$

To form $A(\beta) * x$, we only need the first of these matrices when $\beta = 1.0$, but when $\beta = 0.0$, all 3 are needed. The overhead of reading and using these extra element matrices makes T_M more expensive to use. Moreover, purification must be done explicitly when using T_M; an implicit purification may instead be used when using T_C.

Table 4: Basic operations, times, and residuals for T_C for problem 1 with $Rs = 1975$ using a range of values of r. Implicit shifts of zero and the B semi-inner product are used. $*$ denotes λ_2 was missed.

	$r = 10$	$r = 20$	$r = 25$	$r = 30$	$r = 35$
r_{out}	10	25	25	30	35
Basic operations:					
Factorizations	$*$	5	3	2	2
Linear solves	$*$	119	82	66	76
$A * x$	$*$	90	50	30	35
$B * x$	$*$	339	233	186	216
$(A - \mu B) * x$	$*$	4	2	1	1
Times (secs):					
Factorizations	$*$	35.3	21.0	14.5	14.5
Linear solves	$*$	64.5	44.4	34.9	39.3
$A * x$	$*$	19.2	10.7	6.6	7.7
$B * x$	$*$	80.1	55.6	45.1	53.3
$(A - \mu B) * x$	$*$	1.7	0.8	0.4	0.4
Total	$*$	205.3	135.7	104.6	119.2
Relative residuals:					
Before purification	$*$	1.49d-08	2.13d-06	3.26d-07	3.20d-08
After purification	$*$	1.71d-10	2.43d-08	1.83d-08	8.09d-10

In Table 4 we report results for problem 1 with $Rs = 1975$ for a range of values of r. In this table, r_{out} denotes the final value of r. We see that, even with the test for missing eigenvalues discussed in Section 6.3, with $r = 10$, λ_2 is missed. As r is increased, the number of factorizations (which is equal to one more than the number of Cayley iterations) decreases, until with $r = 30$ the number of factorizations is the minimum possible. This highlights one of the difficulties of using `ARPACK` to solve a problem when the user has no prior knowledge of the spectrum. The user must select r. If r is too small, the sought-after eigenvalues may be missed, and if r is too large, unnecessary work is performed. As discussed earlier, the most expensive parts of the eigenvalue computation in terms of time are those associated with performing matrix factorizations, solving linear systems using the matrix factors, and performing matrix-vector products with $A(\beta)$ and B. Increasing r increases the number of matrix-vector products performed on each iteration. If the number of iterations does not decrease, this can significantly increase the computation time. We see this in Table 4 when we compare $r = 30$ with $r = 35$. If it is important that no eigenvalues are missed, the user should be cautious in the choice of r, and consider checking results by rerunning with an increased value of r and, optionally, increase the number of eigenvalues requested.

In Tables 5 and 6, we present results for problem 1 with $Ra = 2480$ for implicit shifts of zero and exact shifts, both with the B semi-inner product and the standard inner product.

We see that using the B semi-inner product is more reliable than using the standard inner product. With the standard inner product, the modified Cayley transformation T_M failed to compute the left-most eigenvalues. The generalized Cayley transformation T_C was successful in finding the required eigenvalues but, as in the case $Rs = 1975$ discussed above, r was increased to 25 by the test for missing eigenvalues, and this increases the cost of the computation. For this problem, using the B semi-inner product gave the minimum number of restarts; the results were identical for zero and exact implicit shifts. We performed additional experiments with T_C where we did **not** purify the starting vector v_1. We found that, if the B semi-inner product was used, the number of LU factorizations required for convergence increased from 2 to 4 while if the standard inner product was used, the left-most eigenpair was not found. This demonstrates the importance of the choice of starting vector in these calculations.

All the results for problem 1 show the benefits of purifying the computed eigenvectors.

Table 5: Basic operations, times, and residuals for T_C for problem 1 with $Ra = 2480$ and $r = 20$.

	B semi-inner product		Standard inner product	
	Zero shifts	Exact shifts	Zero shifts	Exact shifts
Basic operations:				
Factorizations	2	2	4	4
Linear solves	46	46	98	98
$A * x$	20	20	0	0
$B * x$	127	127	25	25
$(A - \mu B) * x$	1	1	73	73
Times (secs):				
Factorizations	14.3	14.3	27.9	28.0
Linear solves	24.0	25.0	56.6	52.9
$A * x$	4.4	4.2	0.0	0.0
$B * x$	30.9	30.0	5.9	5.9
$(A - \mu B) * x$	0.4	0.4	30.8	30.8
Total	75.7	75.7	124.8	121.3
Relative residuals:				
Before purification	1.19d-08	1.19d-08	2.62d-09	9.29d-09
After purification	5.90d-10	5.90d-10	7.63d-11	4.35d-10

Table 6: Basic operations, times, and residuals for T_M for problem 1 with $Ra = 2480$ and $r = 20$. $*$ denotes λ_1 was missed.

	B semi-inner product		Standard inner product	
	Zero shifts	Exact shifts	Zero shifts	Exact shifts
Basic operations:				
Factorizations	2	2	$*$	$*$
Linear solves	46	46	$*$	$*$
$A(0.0) * x$	20	20	$*$	$*$
$B * x$	127	127	$*$	$*$
$(A(0.0) - \mu B) * x$	1	1	$*$	$*$
Times (secs):				
Factorizations	14.1	14.3	$*$	$*$
Linear solves	24.2	24.8	$*$	$*$
$A(0.0) * x$	13.0	12.8	$*$	$*$
$B * x$	30.2	29.9	$*$	$*$
$(A(0.0) - \mu B) * x$	0.6	0.8	$*$	$*$
Total	83.7	84.3	$*$	$*$
Relative residuals	5.90d-10	5.90d-10	$*$	$*$

9.2 Problem 2

The second problem is that of the flow of a Newtonian fluid past a cylinder in a channel. The problem has 600 elements with a total of 6,398 degrees of freedom. The first bifurcation is a Hopf bifurcation with

$$\lambda_{1,2} = 2.6886 \times 10^{-1} \pm i1.0963 \times 10, \quad \lambda_3 = 2.3827.$$

`MA42` statistics for this problem are given in Table 10. In Tables 8 and 9, we present results for problem 2 with $r = 20$. The convergence tolerance was set to 10^{-6}. Results are given for the generalized and modified Cayley transformations using implicit shifts of zero and exact shifts, both with the B semi-inner product and the standard inner product.

For this problem, using the B semi-inner product does not improve convergence but only adds to the computational cost. Using exact shifts gives slightly slower convergence than implicit shifts of zero.

Table 7: `MA42` statistics for problem 2 ($u = 10^{-7}$).

Flops for factorization	9.99×10^7
Real factor storage (Kwords)	908
Integer factor storage (Kwords)	155
Minimum in-core storage (Kwords)	28
Factorize time (secs)	5.6
Solve time: 1 right-hand side (secs)	0.6
Solve time: 10 right-hand sides (secs)	1.5

Table 8: Basic operations, times, and residuals for T_C for problem 2 with $r = 20$.

	B semi-inner product		Standard inner product	
	Zero shifts	Exact shifts	Zero shifts	Exact shifts
Basic operations:				
Factorizations	4	5	4	5
Linear solves	86	107	86	107
$A * x$	60	80	0	0
$B * x$	222	243	23	23
$(A - \mu B) * x$	3	4	63	84
Times (secs):				
Factorizations	26.4	32.6	26.3	32.5
Linear solves	49.2	65.3	48.8	65.0
$A * x$	26.5	34.2	0.0	0.0
$B * x$	99.2	108.6	10.2	10.2
$(A - \mu B) * x$	2.6	3.4	54.4	71.2
Total	207.5	248.0	143.6	182.8
Relative residuals:				
Before purification	1.23d-03	2.03d-04	1.21d-04	1.86d-05
After purification	9.70d-06	1.51d-06	7.39d-05	1.27d-05

Table 9: Basic operations, times, and residuals for T_M for problem 2 with $r = 20$.

	B semi-inner product		Standard inner product	
	Zero shifts	Exact shifts	Zero shifts	Exact shifts
Basic operations:				
Factorizations	4	5	4	5
Linear solves	86	107	86	107
$A(0.0) * x$	60	80	0	0
$B * x$	220	243	23	23
$(A(0.0) - \mu B) * x$	3	4	63	84
Times (secs):				
Factorizations	26.0	32.7	26.6	32.7
Linear solves	48.3	60.1	48.2	60.0
$A(0.0) * x$	77.0	102.7	0.0	0.0
$B * x$	98.0	108.3	10.2	10.2
$(A(0.0) - \mu B) * x$	4.1	5.2	82.9	109.3
Total	256.8	312.7	171.5	216.2
Relative residuals	1.61d-05	1.48d-05	1.07d-04	9.36d-08

9.3 Problem 3

Our third test problem is that of the flow of a Newtonian fluid in a pipe with sudden symmetric expansion. The problem has 5800 elements and $87,000$ degrees of freedom. The left-most eigenvalue is real but there are several complex eigenvalues λ_j such that $\mathrm{Re}(\lambda_j) - \lambda_1$ is small. In particular,

$$\lambda_1 = 1.962, \quad \lambda_{2,3} = 2.014 \pm i4.118 \times 10^{-1}. \tag{9.2}$$

MA42 statistics for this problem are presented in Table 10. In our experiments, we set $r = 30$ because we found the left-most eigenvalue was missed if we selected a smaller value, such as $r = 20$. For this large problem, while the CPU time required for each LU factorization and each solve is not prohibitive, the additional cost of reading the element matrices from files was found to be high. To limit the time needed to perform each experiment involving this problem, we restricted the maximum number of LU factorizations allowed to 10. With this restriction, the modified Cayley transformation using the B semi-inner product and exact shifts failed to converge. Because of this and the results of our previous experiments, we decided to

Table 10: MA42 statistics for problem 3 ($u = 10^{-7}$).

Flops for factorization	3.54×10^9
Real factor storage (Kwords)	20702
Integer factor storage (Kwords)	2553
Minimum in-core storage (Kwords)	40
Factorize time (secs)	185.2
Solve time: 1 right-hand side (secs)	14.0
Solve time: 10 right-hand sides (secs)	34.0

limit further investigations to using implicit shifts of zero. Results are given in Table 11.

Table 11: Basic operations, times, and residuals for the generalized Cayley transformation ($\beta = 1.0$) and the modified Cayley transformation ($\beta = 0.0$) for problem 3 with $r = 30$. Implicit shifts of zero are used.

	T_C		T_M	
	Inner product		Inner product	
	B	Standard	B	Standard
Basic operations:				
Factorizations	3	4	3	3
Linear solves	95	126	95	95
$A(\beta) * x$	60	0	60	0
$B * x$	271	33	271	33
$(A(\beta) - \mu B) * x$	2	93	2	62
Times (secs):				
Factorizations	588	782	588	586
Linear solves	1352	1736	1310	1304
$A(\beta) * x$	358	0	1102	0
$B * x$	1634	213	1685	212
$(A(\beta) - \mu B) * x$	23	1124	36	1134
Total	4087	4008	4864	3352

9.4 Balancing theory and numerical experiments

We end this section on numerical experiments with some overall comments. Part of the motivation for this report was to numerically verify the theoretical results presented in the paper by Meerbergen and Spence (1997).

As explained in Sections 4.2 and 5.1, care must taken to ensure that the computed eigenvalues and eigenvectors are not contaminated by the $\mathcal{N}$ or $\mathcal{G}$ components that arise in S. In exact arithmetic, this is accomplished by using a starting vector in the range of S^2.

To mitigate the influence that rounding errors might introduce, the scheme proposed by Meerbergen and Spence is to use the B-orthogonal Arnoldi method using one implicit shift of zero per restart and a final implicit purification (via S) of the computed eigenvectors. However, because we use Cayley transformations, their scheme requires us to use an implicit shift equal to $+1$ per restart and to implicitly purify the computed eigenvectors with $T_C - I$.

Here is a summary of our findings:

1. Experiments revealed that the use of the B semi-inner product improved the results. In theory, only $\mathcal{G}$ components in the Arnoldi vectors can contaminate H_r.

2. Using at least one implicit shift equal to $+1$ per restart did not prevent H_r from producing spurious eigenvalues. In theory, this cannot occur because an implicit shift of $+1$ per restart should produce a H_r that is not contaminated by $\mathcal{G}$ components (or $\mathcal{N}$ components that might be present due to rounding errors). We explain why this is so below.

3. $r - s$ implicit shifts of zero per restart gave consistently good results. In theory, this is equivalent to performing subspace iteration with T_C^{r-s} on V_s (per restart). Again, in theory, H_r could be affected by $\mathcal{G}$ components in the Arnoldi vectors.

4. The implicit purification of the computed eigenvectors via $T_C - I$ always decreased the size of the direct residuals.

5. We did not find it necessary to apply T_{SI} a second time to the computed eigenvectors. Theory indicates that the purified eigenvectors may contain $\mathcal{N}$ components arising from a $\mathcal{G}$ component in the unpurified eigenvector.

6. Our results always used a random starting vector in the range of S^2. In theory, this is not needed when using the scheme proposed by Meerbergen and Spence (adapted for Cayley transformations), because of items 3 and 5. We performed some experiments where the starting

vector was not purified. We found that convergence was either significantly slower or the left-most eigenvalue was actually missed (see Section 9.1).

7. It must be emphasized that r must be selected large enough. Except for the possible additional storage, the cost of maintaining the orthogonality of r Arnoldi vectors is not a factor. The cost of the computation is dominated by the cost of factorizing and solving linear systems with A and/or B. Increasing r increases the number of solves which must be performed following a factorization.

Item 2 is easily the most fascinating. The explanation is subtle, but straightforward. From Section 2, an implicit shift equal to $+1$ is equivalent to orthogonalizing the columns of $(T_C - I)V_{r-1}$. By (5.5), this is equivalent to orthogonalizing the columns of $(\sigma - \mu)T_{SI}V_{r-1}$. Thus, if V_{r-1}^+ denotes the updated matrix of Arnoldi vectors produced by the IRAM, the corresponding H_{r-1}^+ contains no spurious eigenvalues. However, as explained in Lehoucq and Sorensen (1996), implicit restarting occasionally undergoes forward instability due to rounding errors. An implicit shift of $+1$ triggers this instability because the Cayley transformation maps the infinite eigenvalues of (1.2) to eigenvalues at $+1$. Because the computed H_r has an eigenvalue equal to $+1$ with an associated eigenvector y, where $e_r^T y$ is small, a small Ritz estimate (see (6.1)) results. This implies that $+1$ is a good approximation for an eigenvalue of T_C. Forward instability results precisely when this nearly converged eigenvalue is used as an implicit shift. Forward instability implies that $\|(T_C - I)V_{r-1} - V_{r-1}^+\| \geq \|(T_C - I)V_{r-1}\|\epsilon_M$. (We remark that this can happen even though the columns of V_{r-1}^+ are orthogonal to machine precision.)

In summary, using implicit shifts equal to $+1$ amplifies any $\mathcal{G}$ or $\mathcal{N}$ components that might be present due to rounding errors when using a Cayley transformation. We remark that Meerbergen and Spence conjectured that spurious eigenvalues could also be computed—even with their scheme. They presented a way to check whether spurious eigenvalues were computed. However, our check (that of using implicit shifts with small Ritz estimates) is cheaper.

10 Conclusions and future directions

We have shown that it is possible to use the implicitly restarted Arnoldi method combined with a generalized Cayley transformation to compute the eigenvalues of the discretized Navier Stokes equations. Our results suggest that although using the B semi-inner product is more expensive than the standard inner product, it does offer advantages in terms of reliability and, in general, gives smaller residuals for the same number of iterations. Because of the connection with subspace iteration, we also found it is more reliable to use zero shifts rather than exact shifts during the implicit restarting used by the IRAM. We have experimented with using a generalized and a modified Cayley transformation. The numerical results for both transformations are similar although, for our examples, use of the modified Cayley transformation was more expensive than the generalized Cayley transformation. The accuracy of the eigenvectors computed using the generalized Cayley transformation can be reduced, sometimes very significantly, by purification with T_{SI}.

Based on our findings, we plan to incorporate the ARPACK software package with the finite-element package ENTWIFE. In the future, we also intend to experiment with a block version of the IRAM (Lehoucq and Maschhoff 1997). Since it is significantly more efficient when using the linear equation solver `MA42` to solve for multiple right-hand sides, we anticipate that this will improve the computation times, particularly for the large problems which are of current interest to AEA Technology.

11 Acknowledgements

This study is part of a larger project to replace the eigensolvers currently used in the finite-element package ENTWIFE by more efficient and robust eigensolvers. We would like to thank Andrew Cliffe of AEA Technology for his interest and for allowing us access to the ENTWIFE code. We are also very grateful to Simon Tavener of Pennsylvania State University for providing us with the test problems for use in this study and for helpful discussions. We thank Gene Golub for some helpful comments at the start of our study, Iain Duff of the Rutherford Appleton Laboratory, Andrew Cliffe, and Simon Tavener for helpful comments on this report, and Karl Meerbergen of Numerical Technologies for some very useful discussions.

References

K.A. Cliffe. ENTWIFE (Release 6.3) Reference Manual. Technical Report AEAT-0823, AEA Technology, Harwell Laboratory, Oxfordshire, England, 1996.

K.A. Cliffe, T.J. Garratt, and A. Spence. Eigenvalues of the discretized Navier-Stokes equation with application to the detection of hopf bifurcations. *Advances in Computational Mathematics*, **1**, 337–356, 1993.

K.A. Cliffe, T.J. Garratt, and A. Spence. Eigenvalues of block matrices arising from problems in fluid mechanics. *SIAM J. Matrix Analysis and Applications*, **15**, 1310–1318, 1994.

J.J. Dongarra, J. DuCroz, I.S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, **16**(1), 1–17, 1990.

I.S. Duff. Design features of a frontal code for solving sparse unsymmetric linear systems out-of-core. *SIAM J. Scientific and Statistical Computing*, **5**, 270–280, 1984.

I.S. Duff and J.A. Scott. MA42 – a new frontal code for solving sparse unsymmetric systems. Technical Report RAL-TR-93-064, Rutherford Appleton Laboratory, 1993.

I.S. Duff and J.A. Scott. A comparison of frontal software with other sparse direct solvers. Technical Report RAL-TR-96-102, Rutherford Appleton Laboratory, 1996*a*.

I.S. Duff and J.A. Scott. The design of a new frontal code for solving sparse unsymmetric systems. *ACM Transactions on Mathematical Software*, **22**(1), 30–45, 1996*b*.

T. Ericsson. A generalised eigenvalue problem and the Lanczos algorithm. *In* J. Cullum and R. A. Willoughby, eds, ‘Large Scale Eigenvalue Problems’, pp. 95–119. Elsevier Science Publications BV, 1986.

T. Ericsson and A. Ruhe. The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, **35**, 1251–1268, 1980.

J.N. Franklin. *Matrix theory.* Prentice-Hall, New Jersey, 1968.

T.J. Garratt. *The numerical detection of Hopf bifurcations in large systems arising in fluid mechanics*. PhD thesis, University of Bath, 1991.

T.J. Garratt, G. Moore, and A. Spence. Two methods for the numerical detection of Hopf bifurcations. *In* R. Seydel, F. W. Schneider and H. Troger, eds, 'Bifurcation and Chaos: Analysis, Algorithms and Applications', pp. 119–123. Birkhauser, 1991.

A. Georgescu. *Hydrodynamic stability analysis*. Martinus Nijhoff, Dordrecht, Netherlands, 1985.

Harwell Subroutine Library. *A Catalogue of Subroutines (Release 12)*. Advanced Computing Department, AEA Technology, Harwell Laboratory, Oxfordshire, England, 1996.

P. Hood. Frontal solution program for unsymmetric matrices. *International Journal on Numerical Methods in Engineering*, **10**, 379–400, 1976.

B.M. Irons. A frontal solution program for finite-element analysis. *International Journal on Numerical Methods in Engineering*, **2**, 5–32, 1970.

R.B. Lehoucq. Truncated QR algorithms and the numerical solution of large scale eigenvalue problems. Preprint MCS-P648-0297, Argonne National Laboratory, Argonne, IL, 1997.

R.B. Lehoucq and K.J. Maschhoff. Implementation of an implicitly restarted block Arnoldi method. Preprint MCS-P649-0297, Argonne National Laboratory, Argonne, IL, 1997.

R.B. Lehoucq and D.C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Analysis and Applications*, **17**(4), 789–821, 1996.

R.B. Lehoucq, D.C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. SIAM, Phildelphia, PA, 1998. To appear.

D.S. Malkus. Eigenproblems associated with the discrete LBB condition for incompressible finite elements. *International Journal for Engineering Science*, **19**, 1299–1310, 1981.

K. Meerbergen and D. Roose. Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems. *IMA Journal of Numerical Analysis*, **16**, 297–346, 1996.

K. Meerbergen and A. Spence. Implicitly restarted Arnoldi with purification for the shift–invert transformation. *Mathematics of Computation*, **218**, 667–689, 1997.

B. Nour-Omid, B.N. Parlett, T. Ericsson, and P.S. Jensen. How to implement the spectral transformation. *Mathematics of Computation*, **48**(178), 663–673, 1987.

J.M. Ortega. *Buoyancy effects in fluids.* C.U.P., Cambridge, England., 1973.

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

B.N. Parlett and Y. Saad. Complex shift and invert strategies for real matrices. *Linear Algebra and Its Applications*, **88/89**(1), 575–595, 1987.

D.H. Sattinger. Transformation groups and bifurcation at multiple eigenvalues. *Bull. Amer. Math. Soc.*, **79**, 709–711, 1973.

D.C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Analysis and Applications*, **13**(1), 357–385, 1992.

G.W. Stewart. *Introduction to Matrix Computations.* Academic Press, San Diego, California, 1973.

D.S. Watkins. *Fundamentals of Matrix Computations.* Wiley, 1991.