JOURNAL OF COMPUTATIONAL PHYSICS 110, 82–102 (1994)

Krylov Methods for the Incompressible Navier–Stokes Equations

W. S. EDWARDS*

Department of Physics and Center for Nonlinear Dynamics, University of Texas at Austin, Austin, Texas 78712

L. S. TUCKERMAN

Center for Nonlinear Dynamics and Department of Mathematics, University of Texas at Austin, Austin, Texas 78712

R. A. FRIESNER

Department of Chemistry, Columbia University, New York, New York 10027

AND

D. C. SORENSEN

Department of Computational and Applied Math, Rice University, Houston, Texas 77251-1892

Received April 20, 1992; revised September 21, 1992

Methods are presented for time evolution, steady-state solving and linear stability analysis for the incompressible Navier–Stokes equations at low to moderate Reynolds numbers. The methods use Krylov subspaces constructed by the Arnoldi process from actions of the explicit Navier–Stokes right-hand side and of its Jacobian, without inversion of the viscous operator. Time evolution is performed by a nonlinear extension of the method of exponential propagation. Steady states are calculated by inexact Krylov–Newton iteration using ORTHORES and GMRES. Linear stability analysis is carried out using an implicitly restarted Arnoldi process with implicit polynomial filters. A detailed implementation is described for a pseudospectral calculation of the stability of Taylor vortices with respect to wavy vortices in the Couette–Taylor problem.

CONTENTS

* Present address: Department of Physics, Haverford College, Haverford, PA 19041.

1. INTRODUCTION

The Navier–Stokes equations pose many challenges to computational physicists. They are three-dimensional nonlinear PDEs, and they are stiff. This stiffness has consequences for the three kinds of calculations typically carried out in computational fluid dynamics—i.e., time evolution, steady-state solving, and linear stability analysis. In this article, we describe and test *Krylov methods* for these three types of calculations in the context of the Couette–Taylor problem, in particular, the transition from Taylor vortices to wavy vortices.

The key concept which unifies our three methods is the use of *Krylov subspaces*. If we write the time-dependent Navier–Stokes equations in the form

$$\partial_t U = F(U) = -(U\cdot\nabla)\,U + \nu\nabla^2 U - \nabla P \qquad (1.1)$$

then our methods rely on explicit evaluations of the function $F(U)$, essentially via the right-hand side of (1.1), and repeated evaluations of its Jacobian $DF(U)$ defined by

$$DF(U)\,u = -(U\cdot\nabla)\,u - (u\cdot\nabla)\,U + \nu\nabla^2 u - \nabla p. \qquad (1.2)$$

0021-9991/94 $6.00

(The handling of the pressure terms ∇P and ∇p to enforce incompressibility, and of boundary conditions, will be explained in the next section.)

Krylov methods for the Navier–Stokes equations are motivated by the following considerations. If we wish to achieve high accuracy at low or moderate Reynolds numbers, computational difficulties arise predominantly from the viscous terms $\nu \nabla^2 U$. The discrete vector Laplacian $\nu \nabla^2$ has a wide range of eigenvalues, which causes the time-dependent nonlinear equations to be *stiff*, and the Jacobian $DF(U)$ to be *poorly conditioned*. This has prompted the development of specialized strategies for each of the three types of computational fluid dynamical calculations, but these strategies generally involve the exact solution of large linear systems.

In time evolution, semi-implicit methods are used, in which the viscous operator $\nu \nabla^2$ (or a similar operator such as $[I - \Delta t \nu \nabla^2]$ for a backwards-Euler scheme) is inverted, directly or iteratively. Direct inversion usually requires a regular geometry and grid in order to take advantage of the concomitant special structure of the Laplacian. However, the constraint of incompressibility, which couples the three velocity components, means that the viscous problem cannot be solved without also solving for the pressure [8–10, 31], except via some approximation. For large problems where construction of the matrix form of the viscous/pressure system is not feasible, exact solution requires an elaborate strategy, such as an influence matrix method [35, 42, 59] to decouple the components and to determine correct pressure boundary conditions. Nonetheless, such methods have been efficiently implemented in many cases.

However, direct inversion of the viscous operator becomes prohibitively expensive for nonstandard geometries or meshes. Iterative inversion—via conjugate gradients, for example—is feasible, but it can converge slowly since the Laplacian becomes increasingly ill-conditioned as the spatial grid is refined. In addition, when the advective term is handled explicitly, as is usually the case, numerical stability is still limited by the CFL condition. Furthermore, the algorithms which immediately suggest themselves for the calculation of steady states (e.g., Newton's method) and for linear stability analysis (the inverse power method), require the manipulation of a matrix, the Jacobian $DF(U)$, whose structure for a general velocity field U is considerably less regular that that of $\nu \nabla^2$.

In most situations, however, it is neither difficult nor expensive to *explicitly* compute the viscous term $\nu \nabla^2 U$ as well as the advective term $-(U \cdot \nabla) U$. In finite difference or finite element methods, the discrete matrix forms of differential operators are sparse, while in the pseudospectral method [4, 26], the action of differential operators can be computed quickly via fast Fourier transforms and recurrence relations. Methods of time evolution, steady-state solving, and linear stability analysis requiring only explicit evaluation and not inversion are, therefore, potentially advantageous.

The idea behind Krylov methods is to use repeated action of the Jacobian $DF(U)$ on some initial velocity field to generate a set of fields that spans a relatively small *Krylov subspace*, in which a good approximate solution of a large linear problem can be found. Such methods are increasingly employed to solve the two major problems of numerical linear algebra: calculating eigenvectors [1, 24, 38, 47, 49, 52, 56, 57] and solving linear equations [48, 50, 51, 61]. A more recent innovation is the use of Krylov methods to solve systems of differential equations [22, 23, 39, 45, 54].

Each Krylov subspace we employ is typically of dimension $K \approx 20$, which means that we require storage for 20 velocity fields with which to span the subspace. Large matrix representations of the Jacobian, however, are not needed; only small $K \times K$ matrices, which approximate its action in the K-dimensional subspace, must be manipulated. Due to their small size, these matrices can be directly diagonalized, exponentiated or inverted as appropriate.

To explain the application of these ideas in the context of the incompressible Navier–Stokes equations, we describe calculations that we have performed to determine the stability of Taylor vortices in the Couette–Taylor problem and to study the instability which gives rise to wavy vortices. Although we limit our study to a pseudospectral discretization, our solution strategies should also be compatible with spatial discretizations based on finite differences, finite elements, or spectral elements. We expect that our methods will be applicable to low to moderate Reynolds number calculations, for which numerical stability poses a major problem, rather than to large-scale turbulence calculations.

In the next section we will describe this linear stability problem, our pseudospectral method, and the operators we use to compute the action of each term in the Navier–Stokes equations. Subsequent sections will describe our time-evolution, steady state, and linear stability methods and explore their efficiency and stability.

2. DESCRIPTION OF THE COUETTE–TAYLOR TEST PROBLEM

The Couette–Taylor problem is a classic hydrodynamic stability problem [17, 58]. An incompressible fluid of kinematic viscosity ν occupies the annular region between long coaxial cylinders. With the outer cylinder at rest and the inner cylinder rotating, the flow undergoes a sequence of transitions as the rotation speed is slowly increased from zero. At the primary transition the basic circular Couette flow bifurcates to an axisymmetric pattern of Taylor vortices. At the secondary transition the Taylor vortex flow bifurcates to a time-dependent rotating-wave state known

as wavy vortices. Our goal is to predict parameter values at which wavy vortices will appear on the basis of a numerical linear stability analysis. Our strategy is to compute the fully nonlinear steady-state axisymmetric Taylor-vortex flow, to linearize the Navier–Stokes equations around this state, and to seek eigenmodes of the linearized equations which break the axisymmetry. Similar calculations have been performed by Jones [32, 33] using other numerical methods.

The radii of the two cylinders are r_{in} and r_{out}, with $r_{out} > r_{in}$. The cylinders are assumed to be infinitely long and the velocity field is periodic in the axial direction with wavelength α. The inner cylinder angular speed is Ω. Three dimensionless numbers, the radius ratio $\eta = r_{in}/r_{out}$, the Reynolds number $\mathrm{Re} = \Omega r_{in}(r_{out} - r_{in})/\nu$, and the dimensionless wavelength $\alpha/(r_{out} - r_{in})$, are sufficient to parameterize the system.

The remainder of this section 2 describes how we spatially discretize the fields and compute the function $F(U)$ and the action of its Jacobian $DF(U)$ as defined in Eqs. (1.1)–(1.2). Because subsequent sections, which describe our solution strategies, make few references to this material, some readers may prefer to bypass it on first reading. Its main importance is to demonstrate how the boundary conditions and divergence-free constraint are incorporated into our methods.

Since this article is primarily concerned with solution strategies for the spatially discretized problem and not with the discretization itself, we will use the notation of the continuum problem to refer also to the spatially discrete operators, for example, using ∇^2 for the discrete Laplacian. Note, however, that the discretization of $F(U)$ is the same as that of U, which is to say that $F(U)$ is computed on the same grid, or expanded in the same set of basis functions, as the velocity field.

2.1. *Geometry and Discretization*

We use a cylindrical coordinate system (r, θ, z). All of our calculations are performed on a two-dimensional (r, z) plane, with $0 \leqslant z < \alpha$, periodic in z, and with $r_{in} \leqslant r \leqslant r_{out}$. The velocity field U has three components (U_r, U_θ, U_z). If $f = f(r, z \bmod \alpha)$ is any one of these three, or the pressure P, then our discrete approximation is the Chebyshev–Fourier series

$$f(r, z) \approx \sum_{n=0}^{N_r} T_n(x(r)) \sum_{j=-N_z/2}^{N_z/2} e^{2\pi i j z/\alpha} f_{nj} + \text{c.c.} \qquad (2.1)$$

Here T_n is the nth Chebyshev polynomial, the function $x(r) \equiv (2r - r_{in} - r_{out})/(r_{out} - r_{in})$ simply maps the interval $r_{in} \leqslant r \leqslant r_{out}$ onto $-1 \leqslant x(r) \leqslant +1$, and the coefficients f_{nj} are real for the Taylor vortex calculation and complex for the wavy-vortex eigenproblem (this latter point will be explained in Section 5). Values of N_r range from 24 to 32 and of N_z from 27 to 72. Conversion to and from a real-space representation is accomplished by mixed-radix fast Fourier transforms. The associated collocation grid is the set of points (r_n, z_j) defined by $x(r_n) = -\cos(\pi n/N_r)$ and $z_j = j\alpha/N_z$. Figure 1 shows a computed Taylor vortex velocity field plotted on this grid.

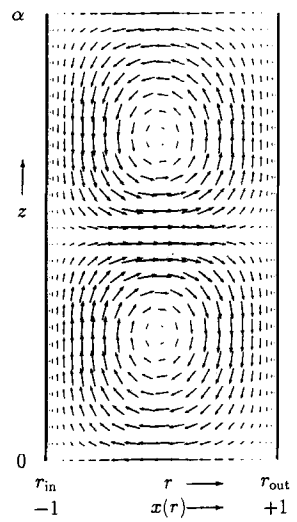

FIG. 1. Computational (r, z) collocation grid and computed nonlinear Taylor-vortex velocity field, for the radius ratio $\eta = 0.8703$, axial wavelength $\alpha = 2.0076(r_{out} - r_{in})$, and Reynolds number $\mathrm{Re} = 131.025$. Each velocity vector refers to the collocation point located at the midpoint of the arrow.

2.2. *Explicit Viscous and Advective Terms*

The incompressible Navier–Stokes equations in cylindrical coordinates are

$$\begin{aligned}
\partial_t U_r &= \{-U_r \partial_r U_r - U_\theta r^{-1} \partial_\theta U_r + U_\theta r^{-1} U_\theta - U_z \partial_z U_r\} \\
&\quad + \nu[\nabla^2 U_r - r^{-2} U_r - 2r^{-2} \partial_\theta U_\theta] - \partial_r P \\
\partial_t U_\theta &= \{-U_r \partial_r U_\theta - U_\theta r^{-1} \partial_\theta U_\theta - U_r r^{-1} U_\theta - U_z \partial_z U_\theta\} \\
&\quad + \nu[\nabla^2 U_\theta - r^{-2} U_\theta + 2r^{-2} \partial_\theta U_r] - r^{-1} \partial_\theta P \\
\partial_t U_z &= \{-U_r \partial_r U_z - U_\theta r^{-1} \partial_\theta U_z - U_z \partial_z U_z\} \\
&\quad + \nu[\nabla^2 U_z] - \partial_z P \\
&\quad r^{-1} \partial_r r U_r + r^{-1} \partial_\theta U_\theta + \partial_z U_z = 0, \qquad (2.2)
\end{aligned}$$

where the scalar Laplacian operator is

$$\nabla^2 = r^{-1} \partial_r r \partial_r + r^{-2} \partial_\theta^2 + \partial_z^2.$$

By convention the partial differential operators ∂_r, ∂_θ, and ∂_z act on everything to their right within the same term, so that, for example, $r^{-1} \partial_r r \partial_r U_r$ is the same as $r^{-1} \partial_r (r \partial_r (U_r))$. We have used braces and square brackets

to indicate the advective $\{-(U\cdot\nabla)\,U\}$ and viscous $[\nu\nabla^2 U]$ terms.

The spatial operators appearing in these equations are the derivatives ∂_r, ∂_θ, ∂_z and the curvature factors r and r^{-1}, as well as product and powers of these operators. For the Taylor vortex calculation, $\partial_\theta \to 0$ because the flow is axisymmetric. In the spectral representation, ∂_z simply multiplies each f_{nj} in (2.1) by $2\pi ij/\alpha$. The operator ∂_r may be evaluated in Chebyshev space via recurrence relations [4, 26]; however, it is often faster to use matrix multiplications for ∂_r, $r^{-1}\,\partial_r r$ and other such strictly radial operators due to the very efficient implementation of small matrix multiplications on many computers. These matrices are of size $(N_r+1)\times(N_r+1)$, with real entries. Since it is equally efficient to act with such a matrix on either the real-space or Chebyshev-space representation, we often choose to store our fields in a mixed spatial representation in which the r dependence is in real space (at the radial collocation points), and the z dependence is spectral (Fourier modes). In this representation the matrices r and r^{-1} are diagonal. No-slip boundary conditions are imposed at collocation points r_{in} and r_{out}.

To evaluate the nonlinear advective terms we compute derivatives in the mixed representation and then convert to real space to perform the multiplications pointwise at each collocation grid point. The result is transformed back to the mixed representation without de-aliasing [4, 26].

2.3. *Pressure and Boundary Conditions*

The no-slip boundary conditions on the cylinder walls at r_{in} and r_{out} are

$$\begin{aligned} &U_r = U_z = 0 && \text{at} \quad r = r_{\text{in}}, r_{\text{out}} \\ &U_\theta = \tilde{\Omega} r_{\text{in}} && \text{at} \quad r = r_{\text{in}} \\ &U_\theta = 0 && \text{at} \quad r = r_{\text{out}}. \end{aligned} \tag{2.3}$$

With Ω constant in time this implies that all three components of $F(U)$ should be set to zero at the boundaries. Let us define the operator B which sets the boundary values of a velocity field to zero. That is, for any U,

$$BU(r,z) = \begin{cases} 0 & \text{at} \quad r = r_{\text{in}}, r_{\text{out}} \\ U(r,z) & \text{elsewhere.} \end{cases}$$

(We note that the discrete form of B for the mixed-space representation of U is simply an $N_r \times N_r$ identity matrix except that the first and last diagonal entries are zero.) The statement that $F(U)$ satisfies the no-slip conditions is equivalent to $F(U) = BF(U)$, that is,

$$\partial_t U = F(U) = B[-(U\cdot\nabla)\,U + \nu\nabla^2 U] - B\nabla P \tag{2.4}$$

which in effect says that the momentum equations are satisfied only in the interior. To derive the *consistent pressure Poisson equation* [8, 10, 27–30], we require that $\nabla\cdot F(U) = 0$ everywhere, including at the walls. From (2.4), we obtain

$$\nabla\cdot B\nabla P = \nabla\cdot B[-(U\cdot\nabla)\,U + \nu\nabla^2 U] \tag{2.5}$$

Equation (2.5) differs from the usual pressure Poisson equation, but which ensures that $\partial_t U$ is (to roundoff error) simultaneously incompressible and zero at the walls; this equation has been used in *explicit* time-stepping algorithms [30, 37]. Articles by Chorin [8–10] and by Gresho and co-authors [27–30] analyze some of the issues surrounding the pressure Poisson equation.

Equation (2.5) requires no explicit boundary conditions for the pressure; they are built-in. For our discretization there is one spurious null mode of the operator $\nabla\cdot B\nabla$ associated with the $j=0$ Fourier mode; however, this problem may be easily bypassed if we note that for this Fourier mode the divergence-free condition (2.2) and the impermeability conditions $U_r = 0$ at $r = r_{\text{in}}, r_{\text{out}}$ together imply that there can be no mean (i.e., $j=0$) radial velocity at any radial position, and thus the $j=0$ radial component of $F(U)$ may simply be set to zero for all r, without ever having computed the $j=0$ component of the pressure. For $j\neq 0$ equation (2.5) is solved by a fast direct method (operation count $O(N_r^2 N_z)$) which takes advantage of the decoupling of the problem by Fourier modes.

To summarize, given a velocity field U we evaluate $F(U)$ by the following steps:

1. Compute and sum together $-(U\cdot\nabla)\,U$ and $\nu\nabla^2 U$.
2. Solve (2.5) for the pressure and subtract its gradient from the result of step 1.
3. Apply the operator B; that is, set the boundary values to zero.

To be more concise, we will say that there exists an idempotent linear operator Π which projects any velocity field U onto a divergence-free field ΠU which is zero on the walls. This operator Π is loosely defined by the expression

$$\Pi U \equiv B[I - \nabla(\nabla\cdot B\nabla)^{-1}\nabla\cdot\,]\,BU, \tag{2.6}$$

where it should be understood that the "inversion" in (2.6) is accomplished by solving the pressure Poisson problem

$$\nabla\cdot B\nabla P = \nabla\cdot BU \tag{2.7}$$

by the fast direct method previously alluded to and with the $j=0$ radial component of the projected field ΠU set to zero

at all r. The projector Π has the desired properties that for any U,

$$\nabla \cdot \Pi U = 0$$

$$[I - B]\, \Pi U = 0,$$

which are the incompressibility and no-slip conditions for the projected field. A single subroutine implements Π in such a manner that P is defined only within the subroutine.

The operator Π allows us to formally remove the pressure from the equations of motion. The right-hand side of (1.1) and the action of the Jacobian defined by (1.2) may now be written concisely as

$$F(U) = \Pi[-(U \cdot \nabla)\, U + \nu \nabla^2 U] \tag{2.8}$$

$$DF(U)\, u = \Pi[-(U \cdot \nabla)\, u - (u \cdot \nabla)\, U + \nu \nabla^2 u]. \tag{2.9}$$

Here we have used the fact that Π is linear and thus has no effect on the linearization of (2.8) to obtain (2.9).

2.4. *Removal of Couette Flow*

Circular Couette flow, which is described by the closed-form expression

$$U_C = U_C(r) = \frac{\Omega r_{\text{in}}^2}{r_{\text{out}}^2 - r_{\text{in}}^2}\left(-r + \frac{r_{\text{out}}^2}{r}\right)\hat{e}_\theta,$$

where $\hat{e}_\theta$ is the azimuthal unit vector, is a steady solution of the full equations which satisfies the boundary conditions (2.3), even though it is an unstable solution at the parameters we are investigating. We find it convenient, however, to redefine U to be the total velocity field minus Couette flow. With this redefinition, $\partial_t(U_C + U) = \partial_t U$ and (2.8) and (2.9) become

$$F(U) = \Pi[-((U_C + U) \cdot \nabla)(U_C + U) + \nu \nabla^2 U] \tag{2.10}$$

$$DF(U)\, u = \Pi[-((U_C + U) \cdot \nabla)\, u - (u \cdot \nabla)(U_C + U) + \nu \nabla^2 u]. \tag{2.11}$$

In (2.10) we have also used the fact that $\nabla^2 U_C = 0$, as can be verified analytically. The newly defined U is subject to the homogeneous boundary conditions

$$U = 0 \qquad \text{at} \quad r = r_{\text{in}}, r_{\text{out}}$$

rather than (2.3). This allows us to use the projector Π to prepare an initial U which satisfies boundary conditions and the divergence-free constraint, for example, by applying Π to a random field.

Our three methods are based on Krylov subspaces spanned by velocity fields made numerically divergence-free by the action of Π. This is not, however, sufficient to prevent the slow accumulation of divergence from round-off error. Thus at the end of each major iteration (e.g., each time step in the time-evolution method, each Newton step in the steady-state method, each restart of the Arnoldi sequence in the linear stability method), we apply the projector once more to each velocity field that is carried forward to the next major iteration. This is a subtle point which perhaps will not become clear until the various methods are discussed more fully.

2.5. *Summary*

The subroutines which compute $F(U)$, $DF(U)\, u$, and ΠU are elementary fluid dynamical routines exercised by our various Krylov methods. An additional subroutine, $G(u)$, is used in time evolution, and it will be defined and described in Section 3. A modified Jacobian $DF_{mq}(U)\, u$ and a modified projector $\Pi_{mq} U$ are used in the linear stability calculations, where m and q refer to the azimuthal wavenumber and axial Floquet exponent, respectively, of the wavy vortex eigenmode; this will be explained in Section 5. For completeness these six subroutines are summarized in Table I. Essentially all information related to the fluid dynamical problem is contained within these subroutines, which are used as black boxes by the driver

TABLE I

The Elementary Subroutines Used by Our Krylov Methods

Name	Description	Definition	CPU
$F(U)$	Right-hand side	$\Pi[-((U_C + U) \cdot \nabla)(U_C + U) + \nu \nabla^2 U]$	0.013
$DF(U)\, u$	Jacobian	$\Pi[-((U_C + U) \cdot \nabla)\, u - (u \cdot \nabla)(U_C + U) + \nu \nabla^2 u]$	0.014
ΠU	Projector	$B[U - \nabla P]$, where $\nabla \cdot B \nabla P = \nabla \cdot BU$	0.003
$G(u)$	Remainder	$\Pi[-(u \cdot \nabla)\, u]$	0.009
$DF_{mq}(U)\, u$	Wavy vortex Jacobian	$\Pi_{mq}[-((U_C + U) \cdot \nabla_{mq})\, u - (u \cdot \nabla)(U_C + U) + \nu \nabla_{mq}^2 u]$	0.014
$\Pi_{mq} u$	Wavy vortex projector	$B[u - \nabla_{mq} p]$, where $\nabla_{mq} \cdot B \nabla_{mq} p = \nabla_{mq} \cdot Bu$	0.003

Note. All information about the fluid dynamical problem, its boundary conditions, the divergence-free constraint, and the spatial discretization, is contained within these routines. They are treated essentially as black boxes by the various drivers that implement our solution strategies. CPU timings are for a single Cray YMP processor and truncation parameters $N = 24$ and $J = 27$. Total CPU usage shown in figures is dominated by actions of $DF(U)\, u$; hence estimates of the total number of invocations of $DF(U)\, u$ are obtained with the conversion factor 0.014 CPU seconds per subroutine call.

routines which implement our solution strategies. The three computational fluid dynamical problems to be solved in the remainder of this article are written succinctly in terms of these functions and operators as

1. the time-dependent nonlinear problem for Taylor vortices

$$\partial_t U = F(U)$$

2. the steady-state problem for Taylor vortices

$$F(U) = 0$$

which leads to the Newton iteration

$$DF(U)\, u = F(U)$$

and

3. the linear stability eigenproblem for wavy vortices

$$DF_{mq}(U)\, u = \lambda u.$$

3. TIME EVOLUTION BY EXPONENTIAL PROPAGATION

The first major computational fluid dynamical problem we address is time evolution. One seeks to obtain the time history of the evolving velocity field $U(t)$ by solving the time-dependent problem $\partial_t U = F(U)$ beginning with some suitable initial condition $U(0)$. In this section 3 we describe and test a method of time evolution based on the use of Krylov subspaces. We begin with the case of a homogeneous linear system in order to set notation and establish basic ideas and results. We describe how an approximation to the exponential of a large linear operator can be constructed using the exact exponential of a small matrix which approximates the action of the operator in a K-dimensional Krylov subspace. Use of this approximate exponential allows integration from time t_0 to $t_0 + t$ with $O(t^K)$ accuracy. Furthermore, in our numerical experiments with the Navier–Stokes operators, the method appears to be numerically stable for all t (although we cannot prove this in general). We then extend the method to the nonlinear case, incorporating nonlinear corrections via a functional iteration method. We test the nonlinear time-evolution method using $F(U)$ and $DF(U)$ as defined for the Couette–Taylor problem. Our test case will be the evolution of the axisymmetric Taylor-vortex flow from a suitable initial condition to the time-asymptotic steady state. After presenting our timings, we will compare the Krylov methods to standard methods of time evolution.

3.1. *Exponential Propagation for Linear Differential Equations*

Consider the linear initial-value problem:

$$\partial_t U = AU \tag{3.1}$$

$$U(t_0) = U_0 \tag{3.2}$$

for an arbitrary linear operator A. The exact solution of (3.1)–(3.2) is

$$U(t_0 + t) = e^{tA} U_0. \tag{3.3}$$

Suppose that $U(t)$ is represented in discrete form by N unknowns. An operator such as tA may be exponentiated if the $N \times N$ matrix form of A can be decomposed as

$$A = E \Lambda E^{-1} \tag{3.4}$$

where E is a matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues $\lambda_1, \lambda_2, ..., \lambda_N$. In this case

$$e^{tA} = E e^{t\Lambda} E^{-1}, \tag{3.5}$$

where $e^{t\Lambda}$ is the diagonal matrix with entries $e^{t\lambda_1}, e^{t\lambda_2}, ..., e^{t\lambda_N}$. However, exponentiation of an operator A via exact diagonalization of its matrix form is quite expensive if N is large. For the discretization defined for our Couette–Taylor problem, the number of unknowns in the expansions (2.1) of the three velocity components is $N \equiv (N_r + 1) \times N_z \times 3$, which is 2025 for the typical values $N_r = 24$ and $N_z = 27$. We will often refer to the discrete representation of the velocity field U as a *vector* of length N, by which we mean a column vector of the expansion coefficients arranged in some fixed but otherwise arbitrary storage order. The matrix A would then be of size $N \times N$. Even if such a large amount of storage were available, the computational cost of the diagonalization step (3.4) would make direct use of (3.5) uncompetitive with standard implicit multistep methods. Instead, we shall approximate the action of e^{tA} on an initial vector U_0 by exponentiating a small matrix H which approximates the action of A in a Krylov subspace.

We define the K-dimensional Krylov subspace to be the space spanned by the set of vectors $\{U_0, AU_0, ..., A^{K-1}U_0\}$. An orthonormal basis for this subspace is generated by the following *Arnoldi process* [1]. Setting $w_1 \equiv U_0$, we compute, for $k = 1, 2, ..., K$,

$$v_k \equiv w_k / \|w_k\| \tag{3.6}$$

$$w_{k+1} \equiv Av_k - \sum_{l=1}^{k} v_l (v_l, Av_k). \tag{3.7}$$

This is equivalent to the Gram–Schmidt orthogonalization of the sequence of vectors $\{U_0, AU_0, ..., A^{K-1}U_0\}$ and the

K resulting *Krylov vectors* $v_1, v_2, ..., v_K$ are orthonormal vectors in $\mathscr{R}^N$. Although finite precision arithmetic can sometimes lead to loss of orthogonality, we find empirically that for our Couette–Taylor calculations, straightforward implementation of (3.6)–(3.7) results in an acceptably orthogonal basis set.

We note that the Arnoldi process requires only the *action* of the linear operator A on each vector v_k. Thus in the actual implementation it is sufficient to have available a subroutine which computes this action, by whatever means, but it is not necessary to have available any matrix representation of A. In particular, the Jacobian $DF(U)$ for our Couette–Taylor problem is a subroutine which computes the action of a linear operator. We note that only K actions with the operator A (that is, K subroutine calls) are sufficient to carry out the K steps of the Arnoldi process. Recall that K will be approximately 20 for our tests, so that the Krylov subspaces have dimension very much less than $N = 2025$, the typical problem size defined by the spatial discretization.

Now that we have constructed the orthonormal basis, we assemble the Krylov vectors into an $N \times K$ matrix V whose columns are $v_1, v_2, ..., v_K$. That is,

$$Ve_k \equiv v_k, \qquad k = 1, ..., K,$$

where e_k is the kth unit vector of length K. Storage requirements for our methods are dominated by the size of V, which is the largest matrix that we explicitly construct. We note that the operator VV^{T}, where V^{T} is the transpose of V, is an $N \times N$ projection operator onto the Krylov space, and that $V^{\mathrm{T}}V$ is the $K \times K$ identity matrix.

The action of the operator A may be approximated in the Krylov subspace by

$$A \approx VV^{\mathrm{T}}AVV^{\mathrm{T}} = VHV^{\mathrm{T}},$$

where the $K \times K$ matrix

$$H \equiv V^{\mathrm{T}}AV \tag{3.8}$$

has elements

$$H_{lk} \equiv (v_l, Av_k). \tag{3.9}$$

By virtue of definition (3.6)–(3.7), v_l is orthogonal to Av_k for $l > k+1$; thus H is upper Hessenberg. The scalar products that are subtracted from Av_k to form w_{k+1} in (3.7) comprise most of the matrix elements of H. It can also be verified that $H_{k+1,k} = \|w_{k+1}\|$, so that the entire matrix H is generated automatically as a by-product of orthogonalization in the Arnoldi process.

It will prove convenient for later sections to restate Eq. (3.7) in the standard matrix form of the Arnoldi decomposition,

$$AV = VH + we_K^{\mathrm{T}}, \tag{3.10}$$

where $w \equiv w_{K+1}$. Equation (3.10) states that the product of A with any Krylov vector, except the last one, is in the Krylov subspace. Here, A is $N \times N$, V is $N \times K$, H is $K \times K$, w is $N \times 1$, and e_K^{T} is $1 \times K$, so that each term in (3.10) is $N \times K$. We show that (3.10) holds by showing that each column of the left-hand side equals the corresponding column of the right-hand side; that is,

$$AVe_k = (VH + we_K^{\mathrm{T}})\, e_k, \qquad k = 1, ..., K. \tag{3.11}$$

For the case $k = K$, this becomes

$$Av_K = \sum_{l=1}^{K} v_l H_{lK} + w$$

which follows from (3.7) and (3.9). For $1 \leqslant k < K$, (3.11) becomes

$$\begin{aligned} Av_k &= \sum_{l=1}^{K} v_l H_{lk} \\ &= \sum_{l=1}^{k} v_l H_{lk} + v_{k+1} H_{k+1,k} \qquad (3.12) \\ &= \sum_{l=1}^{k} v_l H_{lk} + w_{k+1} \qquad (3.13) \end{aligned}$$

which also follows from (3.7). In (3.12) we have used the fact that H is upper Hessenberg and in (3.13) we have used $H_{k+1,k} = \|w_{k+1}\|$ and then (3.6) for $k+1$.

In order to approximate the exponential of tA, we now diagonalize the small matrix H explicitly, for example by the QR algorithm, to yield

$$H = E\Lambda E^{-1},$$

where Λ is a diagonal $K \times K$ matrix of eigenvalues $\lambda_1, \lambda_2, ..., \lambda_K$ and E is the $K \times K$ matrix whose columns are the corresponding eigenvectors. The solution (3.3) at time $t_0 + t$ is then approximated by

$$\begin{aligned} U(t_0 + t) &= e^{t\mathscr{A}}U_0 \approx Ve^{tH}V^{\mathrm{T}}U_0 \qquad (3.14) \\ &= VEe^{t\Lambda}E^{-1}V^{\mathrm{T}}U_0, \qquad (3.15) \end{aligned}$$

where $e^{t\Lambda}$ is the diagonal matrix with entries $e^{t\lambda_1}, e^{t\lambda_2}, ..., e^{t\lambda_K}$. In Eq. (3.14), the product $V^{\mathrm{T}}U_0$ need not actually be calculated, since U_0 is used to *generate* the Krylov space.

That is, $U_0 = v_1 \|U_0\|$, so

$$V^T U_0 = V^T v_1 \|U_0\| = e_1 \|U_0\|,$$

where v_1 is the first Krylov vector and e_1 is the first unit vector of length K. We note that in our numerical experiments the cost of explicitly diagonalizing and exponentiating H is always much smaller than the cost of the K steps of the Arnoldi sequence when A is $DF(U)$.

The error in approximation (3.14) can be shown to be of order $O(t^K)$ by the following calculation:

$$\begin{aligned} Ve^{tH}V^T U_0 &= V\left(I + tH + \frac{t^2}{2}H^2 + \cdots\right) V^T U_0 \\ &= V\left(I + tV^T AV + \frac{t^2}{2}V^T AVV^T AV + \cdots\right) V^T U_0 \\ &= \left(I + tVV^T A + \frac{t^2}{2}VV^T AVV^T A + \cdots\right) VV^T U_0. \end{aligned} \tag{3.16}$$

The operator VV^T which appears repeatedly in (3.16) is the projection operator onto the Krylov space. Again, since U_0 generates the Krylov space, it is in the Krylov space and $VV^T U_0 = U_0$. Similarly, AU_0 is in the Krylov space, so $VV^T AU_0 = AU_0$. This reasoning can be continued up until the term $VV^T A^{K-1}U_0 = A^{K-1}U_0$. The end result is

$$\begin{aligned} Ve^{tH}V^T U_0 = &\left(I + tA + \frac{(tA)^2}{2} + \cdots + \frac{(tA)^{K-1}}{(K-1)!}\right) U_0 \\ &+ \left(\frac{t^K}{K!}I + \frac{t^{K+1}}{(K+1)!}VV^T A + \cdots\right) \\ &\times VV^T A^K U_0 \end{aligned} \tag{3.17}$$

which approximates e^{tA} to order t^K. A more rigorous proof and upper bound for the error can be found in [23, 53].

Having discussed the accuracy of (3.14), we now address its stability. It can be shown that (3.14) is *unconditionally stable* when A is symmetric. By unconditionally stable we mean that if

$$\lim_{t \to \infty} \|e^{tA}U_0\| = 0$$

then

$$\lim_{t \to \infty} \|Ve^{tH}V^T U_0\| = 0.$$

The above is equivalent to the statement that if the operator A is *stable* (i.e., if all of its eigenvalues have negative real parts), then the operator H is also stable.

The proof follows from the fact that, for symmetric matrices, stability and negative definiteness are synonymous. It is easy to show that if A is symmetric and negative definite, then so is H [21, 46]. The symmetry of H follows simply from

$$H^T = (V^T AV)^T = V^T A^T V = V^T AV = H.$$

The matrix H is negative definite since, for any $y \neq 0$ in $\mathbb{R}^K$,

$$\begin{aligned} (y, Hy) &= (y, V^T AVy) \\ &= (Vy, AVy) < 0. \end{aligned}$$

This proof can be extended to apply to *normal* operators, i.e., $A^T A = AA^T$. That is, if A is normal, then stability of A implies stability of H. Gallopoulos and Saad [23] also prove that if the *symmetric part* of A (i.e., $(A + A^T)/2$) is stable, then so is the symmetric part of H. For general nonsymmetric A, however, unconditional stability cannot be proven, and in fact there exist stable operators A for which certain starting vectors U_0 generate unstable matrices H.

Although the Navier–Stokes Jacobian $DF(U)$ is neither symmetric nor normal, for very low Reynolds numbers it is dominated by the Stokes operator $\Pi\nu\nabla^2$. The Stokes operator is symmetric negative definite for certain problems, for example, a centered finite-difference scheme in Cartesian coordinates with periodic boundary conditions. For such a Stokes problem, therefore, unconditional stability can be shown rigorously. Furthermore, as the Reynolds number is increased from zero, the eigenvalues of $DF(U)$ depart continuously from the Stokes eigenvalues, and thus for simple problems there must exist *some range of Reynolds numbers* for which the unconditional stability of (3.14) is still in effect. (We note also that no straightforward analog of the CFL stability restriction is apparent.)

For the Couette–Taylor problem the Stokes operator $\Pi\nu\nabla^2$ as defined in cylindrical coordinates for the Chebyshev–Fourier basis is symmetric *only with respect to a certain inner product*, which is not the standard Euclidean inner product used in our Arnoldi process. Thus we know of no proof of the unconditional stability of (3.14) for the operator $DF(U)$ corresponding to our discretization of, and choice of inner product for, the Couette–Taylor problem (no matter how small the Reynolds number). In our numerical experiments, however, unconditional stability is often evident. We will return to this issue in Section 3.5 when we discuss our timings.

It may be noted that since exponential propagation approximates $U(t_0 + t)$ within a Krylov space spanned by $A^k U_0$, $k = 0, \ldots, K-1$, the result is a linear combination of these vectors and must therefore be the sum of a certain K-term series in powers of A acting on U_0. Series approximations to the exponential of tA are also used by

explicit Taylor series methods, which are well known to be unstable at large t for the simple reason that polynomials are unbounded for sufficiently large arguments. In view of this, the unconditional stability of exponential propagation for the Stokes operator may seem surprising. The essential difference, however, is that explicit formulas depend on the product tA, so that the coefficient of $A^k U_0$ has the functional form t^k. In contrast, in exponential propagation (3.14), the coefficient of each term $A^k U_0$ is determined from the matrix A and the initial condition U_0 via the Arnoldi process and the exponentiation of tH, and thus is not of the form t^k. Thus the series approximation to the exponential may remain numerically stable unconditionally, as is in fact the case when all eigenvalues of H have negative real parts.

For fixed K, the error in approximation (3.14–3.15) increases with t. In order to solve (3.1–3.2) to fixed accuracy for a given t, it is therefore necessary to incorporate (3.14–3.15) into a time-stepping method. A method of time-evolution based on exponential propagation via (3.15) was first suggested in [39, 45] for symmetric linear systems arising in the context of certain problems in computational chemistry and later adapted for nonsymmetric systems of differential equations in [22, 23]. Starting with an initial vector $U_0 = U(t_0)$, one obtains a vector $U_1 \approx U(t_0 + t)$ for some time step t by carrying out the Arnoldi process, the construction and exponentiation of H and the computation of U_1 via (3.14). The vector U_1 then becomes the initial vector for the next time step, and it is used to generate a new Krylov subspace and to construct updated matrices H and V. Thus the approximate representation of the matrix A changes at each time step. The size t of the time step is determined by accuracy requirements and is varied at each step by an automatic error-control mechanism.

The basic difference between solving linear systems by exponential propagation and by standard multistep methods can be summarized as follows. Multistep methods *approximate the exponential* (e.g., by a polynomial as in forwards Euler, or by a rational function as in backwards Euler and Crank–Nicolson) of the *exact* operator tA. Exponential propagation instead uses the *exact exponential* of the *approximate* operator $tVHV^{\mathrm{T}}$.

3.2. *Exponential Propagation for Nonlinear Differential Equations*

Exponential propagation has been adapted for solving nonlinear systems of differential equations by Friesner *et al.* [22]. Consider the general nonlinear initial-value problem:

$$\partial_t U = F(U)$$

$$U(t_0) = U_0.$$

By defining

$$u(t) \equiv U(t_0 + t) - U_0 \tag{3.18}$$

and Taylor expanding F about U_0, we obtain the problem for u,

$$\partial_t u = b + Au + G(u) \tag{3.19}$$

$$u(t_0) = 0, \tag{3.20}$$

where $b \equiv F(U_0)$, the Jacobian $A \equiv DF(U_0)$, and the remainder $G(u)$ is defined by

$$G(u) \equiv F(U_0 + u) - b - Au. \tag{3.21}$$

For our fluid dynamical problem, $F(U_0)$ and $DF(U_0)$ are as defined in (2.10)–(2.11) and Table I, and

$$G(u) = \Pi[-(u \cdot \nabla) u] \tag{3.22}$$

as can be verified from (3.21) and the definitions of $F(U)$ and $DF(U)$. The action of $G(u)$ is computed by an elementary fluid dynamical subroutine, similar to the subroutines $F(U)$ and $DF(U)$ previously described. The simple form of expression (3.22) results from the straightforward quadratic nonlinearity of the Navier–Stokes equations. (For other equations with more difficult nonlinearities, it would be necessary to use the original expression (3.21).)

If $G(u)$ were zero, then the exact solution to (3.19)–(3.20) would be

$$u(t) = \frac{e^{tA} - I}{A} b. \tag{3.23}$$

There is, of course, no closed form solution to (3.19)–(3.20) when $G(u) \neq 0$. But we may write the equivalent integral equation:

$$u(t) = \frac{e^{tA} - I}{A} b + \int_0^t d\tau\, e^{(t-\tau)A} G(u(\tau)). \tag{3.24}$$

We solve (3.24) by the functional iteration

$$u^{(m+1)}(t) = \frac{e^{tA} - I}{A} b + \int_0^t d\tau\, e^{(t-\tau)A} G(u^{(m)}(\tau)), \tag{3.25}$$

using (3.23) as the initial guess $u^{(0)}(t)$. Equation (3.23) is approximated, by analogy to (3.15) as

$$u^{(0)}(t) \approx VE \frac{e^{tA} - I}{A} E^{-1} V^{\mathrm{T}} b. \tag{3.26}$$

Recall that A is diagonal, so that its inversion and exponentiation is straightforward. Expression (3.26) can be shown to be an $O(t^{K+1})$ accurate approximation to (3.23) by a calculation analogous to that leading to (3.17).

The convolution integral in (3.25) is evaluated by a combination of linear exponential propagation and quadrature, as follows. We calculate $G(u^{(m)}(\tau))$ at several values of τ in order to fit $G(u^{(m)}(\tau))$ to a polynomial,

$$G(u^{(m)}(\tau)) \approx \sum_{j=2}^{J+1} g_j \tau^j, \tag{3.27}$$

where each g_j is a vector of length N. The polynomial has no constant or linear terms in τ by virtue of the definition (3.21) of the remainder $G(u)$. Typically the degree of the polynomial is fixed at three, so that only two values of $\tau > 0$ are needed and the polynomial (3.27) has only $J = 2$ terms.

The integral in (3.25) can now be written as

$$\int_0^t d\tau\, e^{(t-\tau)A} G(u^{(m)}(\tau))$$

$$\approx \int_0^t d\tau\, e^{(t-\tau)A} \sum_j g_j \tau^j \tag{3.28}$$

$$= \sum_j \left[\int_0^t d\tau\, e^{(t-\tau)A_j} \tau^j \right] g_j. \tag{3.29}$$

We act with the integral operator in square brackets in (3.29) on each of the vector coefficients g_j by forming a separate Krylov subspace V_j and approximate operator H_j for each j. This involves carrying out the Arnoldi process using g_j as the initial vector. However, here we find empirically that much smaller Krylov spaces ($K_j \approx 2$) are optimal. After diagonalizing each small matrix $H_j = E_j \Lambda_j E_j^{-1}$, the final approximation for the convolution integral (3.28) is assembled as

$$\int_0^t d\tau\, e^{(t-\tau)A} G(u^{(m)}(\tau))$$

$$\approx \sum_j V_j E_j \left(\int_0^t d\tau\, e^{(t-\tau)\Lambda_j} \tau^j \right) E_j^{-1} V_j^{\mathrm{T}} g_j, \tag{3.30}$$

where the Λ_j are $K_j \times K_j$ diagonal matrices. The integrals in (3.30) enclosed in parentheses are evaluated in closed form via a recursion relation. We find that the functional iteration (3.25) via (3.26), (3.27), and (3.30) converges typically in $m = 2$ iterations. Although expression (3.30) looks computationally formidable, no purpose would be served in evaluating the integral more accurately than the linear term (3.26); it is for this reason that we use the minimal value of two for each of J, K_j, and m, since we find that the error in evaluating (3.30) is then comparable to the error in evaluating (3.26) with $K = 20$.

For each value of t, the final functional form for $u(t)$ is set to be the sum of (3.26) and (3.30):

$$u(t) = VE \frac{e^{t\Lambda} - I}{\Lambda} E^{-1} V^{\mathrm{T}} b + \sum_j V_j E_j$$

$$\times \left(\int_0^t d\tau\, e^{(t-\tau)\Lambda_j} \tau^j \right) E_j^{-1} V_j^{\mathrm{T}} g_j. \tag{3.31}$$

3.3. *Automatic Error Control*

Equation (3.31) can be used to approximate $u(t)$, and thus to recover $U(t_0 + t)$ via (3.18), for various values of t. Our automatic error control technique makes use of the observation that the *time derivative* of this expression may also be evaluated *at negligible cost* for various t's, as a by-product of the computations performed during the final functional iteration step. Since the exact time derivative may also be obtained by evaluating $F(U(t_0 + t))$ with a subroutine call, a reliable measure for the error in the time derivative is obtained. We may then select a posteriori, that is, after the final functional iteration step, the largest value of t for which the solution is acceptable according to a preestablished accuracy criterion. The solution $U(t_0 + t)$ for this maximum t is carried forward to become the initial condition for the next time step.

The evaluation of the time derivative of (3.31) is performed as follows. The equation can be differentiated analytically to yield

$$\partial_t u(t) = VEe^{t\Lambda} E^{-1} V^{\mathrm{T}} b + \sum_j V_j E_j \Lambda_j$$

$$\times \left(\int_0^t d\tau\, e^{(t-\tau)\Lambda_j} \tau^j \right) E_j^{-1} V_j^{\mathrm{T}} g_j$$

$$+ \sum_j V_j E_j t^j E_j^{-1} V_j^{\mathrm{T}} g_j. \tag{3.32}$$

The first two terms in the above expression are easily evaluated at various t's as by-products of the evaluation of the linear solution (3.26) and the integral (3.30). Referring to Eq. (3.27), the last term in (3.32) is seen to be merely $G(u(t))$, since

$$V_j E_j t^j E_j^{-1} V_j^{\mathrm{T}} g_j = V_j V_j^{\mathrm{T}} g_j t^j = g_j t^j.$$

(Recall that $V_j V_j^{\mathrm{T}}$ projects onto the Krylov subspace which, in this case, contains g_j because the subspace is generated by starting the Arnoldi sequence with g_j.)

Given values for $u(t)$, we can also compute $F(U_0 + u(t))$ and compare this with the evaluation of Eq. (3.32). We define a residual field via

$$\Delta u(t) \equiv t[\partial_t u(t) - F(U_0 + u(t))]. \tag{3.33}$$

In test cases for which closed-form solutions were available [22], it was found that the residual field Δu calculated via (3.33) corresponded well to the actual error in u.

As a scalar criterion for acceptable accuracy, we have chosen to use the relative error tolerance condition

$$\frac{\|\Delta u(t)\|_\infty}{\|U_0\|_2} < \varepsilon, \tag{3.34}$$

where

$$\|u\|_\infty \equiv \max_i |u_i| \tag{3.35}$$

$$\|u\|_2 \equiv \left(\sum_i u_i^2\right)^{1/2}, \tag{3.36}$$

where i ranges from 1 to N. Norms other than (3.35)–(3.36) have been found to yield comparable results.

3.4. *Taylor Vortex Test Computations*

Figure 2 shows the time evolution of the axisymmetric Taylor vortex problem for a specific set of parameters. The initial small-amplitude velocity field (a) was produced by solving the linear stability problem for Couette flow at the same parameters and selecting the leading eigenmode. (This two-point boundary eigenproblem in r was solved by a straightforward method described in [19, 36, 40, 42, 44].) The initial field is sinusoidal in z and vortex centers are equidistant from inflow and outflow jets, which are of equal strength. The faint vortex structure grows until its amplitude approaches an asymptotic limit. This nonlinearly saturated flow (c) has vortex centers noticeably displaced toward the outflow jet, which is stronger and more localized than the inflow jet. A transient velocity field, which we will use in Section 4 to initiate our Krylov–Newton method, is shown in (b).

Figure 3 shows the expenditure of CPU time for the time evolution of Fig. 2. The horizontal axis displays what we call the model time, i.e., the fluid dynamical time in units of the viscous diffusion time $(r_{out}-r_{in})^2/\nu$. The CPU time per model time, i.e., the slope of the curve, becomes very small as the asymptotic steady state is approached. In this regime, the automatic error control algorithm described by Eqs. (3.33)–(3.34) determines that increasingly large timesteps t are allowable without exceeding the specified relative error tolerance ε. The selected timesteps increase apparently without bound. This demonstrates the unusual stability of the exponential propagation method, since the very large negative eigenvalues of the Jacobian $DF(U)$, due to $\nu\nabla^2$, persist regardless of U. In this regime we have found that the eigenvalues of the Hessenberg matrix H used in the evaluation of (3.26) have negative real parts, so that (3.26) is in fact unconditionally stable. Thus we see that the allowable timesteps are determined exclusively by the accuracy requirements, which is the hallmark of a stiff-equation solver.

Figures 4 and 5 explore the efficiency of the method as the two most important parameters, K and ε, are varied. Figure 4 varies the Krylov-space dimension K, keeping ε fixed at 10^{-4}; we learn that there is a value of K which results in the least expenditure of CPU time for any given evolution time of the model and that this value is about 20. We also see again that the efficiency of time evolution increases apparently without bound as the steady-state

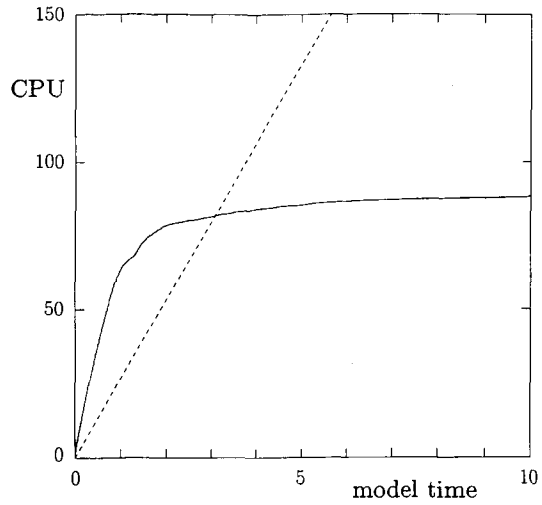

FIG. 3. Efficiency of time evolution. The CPU time required to evolve the axisymmetric Taylor vortex flow is plotted as a function of model time in units of the viscous diffusion time $(r_{out}-r_{in})^2/\nu$. These results are for $K=20$ and $\varepsilon=10^{-4}$. Each time step of the exponential propagation method consumes an approximately constant amount of CPU time, which for this case is about 0.43 s; 196 such time steps were required for this run. The increment of model time t accomplished by each step increases as the steady-state solution is approached, apparently without bound. The variation in t results from the adaptive error control technique, which at each time step seeks to maximize t while keeping the solution at all times within the relative error tolerance of 10^{-4}. The dashed line is an estimate of the performance that would be achieved by a semi-implicit multistep method with fixed timestep t.

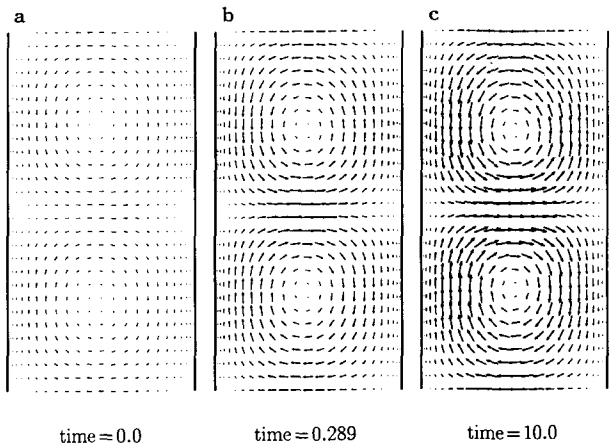

FIG. 2. Time evolution of the Taylor vortex flow of Fig. 1: (a) shows the initial condition; (b) shows the field at time = 0.289; and (c) shows the asymptotic field at time = 10. Times are in units of the viscous diffusion time defined by $(r_{out}-r_{in})^2/\nu$. The field in (b) will be used as the initial condition for our steady-state solver in the following section; (c) is the same as Fig. 1.

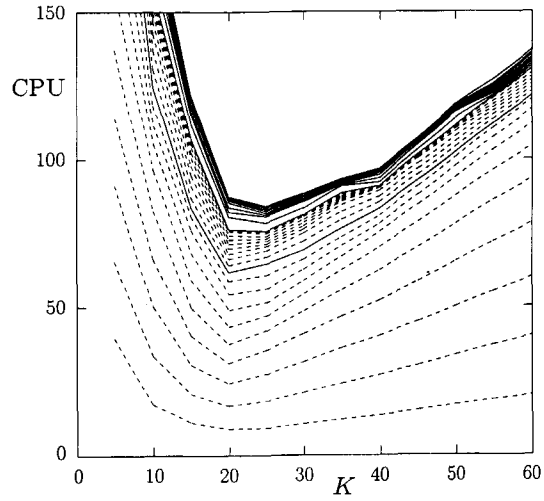

FIG. 4. Efficiency of time evolution versus the dimension K of the Krylov subspaces used in the exponential propagation method. The plot summarizes the results of twelve runs with $K = 5, 10, 15, ..., 60$, for the initial condition of Fig. 2a. Each curve is a contour of constant model time. The dashed contours are for times 0.1, 0.2, 0.3, ... in units of $(r_{out} - r_{in})^2/\nu$. The solid contours are for times 1, 2, 3, ..., 10. The accumulation of the solid curves indicates that the time steps t used by the method are increasing, apparently without bound, as the steady-state solution is approached. The relative error tolerance was $\varepsilon = 10^{-4}$ for all runs.

solution is approached, as shown by the accumulation of the contours of model time as is it approaches 10. Figure 5 varies the relative error tolerance ε, keeping K fixed at 20. CPU timings are for a single Cray YMP processor. Vectorization occurs primarily in the elementary hydrodynamic subroutines over one or both spatial directions (N_r and/or N_z, which are fixed) and thus does not effect the dependence of CPU time on K.

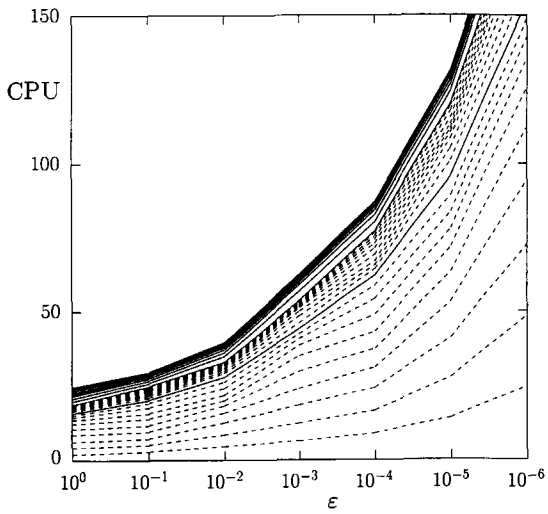

FIG. 5. Efficiency of time evolution versus relative error tolerance. The dashed and solid lines are as defined in Fig. 4. Six runs were performed, with relative error tolerances $10^{-1}, 10^{-2}, ..., 10^{-6}$.

3.5. *Comparison with Other Methods*

Clearly there are many differences between our Krylov method for time evolution and other, more common methods. Let us compare our method to typical semi-implicit methods.

1. *Accuracy.* The semi-implicit methods usually achieve only up to second-order formal accuracy in the timestep t, as there is no unconditionally stable third-order implicit multistep method. The formal order of accuracy of the Krylov method is known only for the homogeneous linear system, by the calculation leading to Eq. (3.17), but not for the nonlinear case. The Krylov method incorporates an automatic error control technique. Semi-implicit schemes can also be implemented with automatic error control, but this requires the ability to vary t during the calculation. Many semi-implicit codes use fixed t, however, because this allows preprocessing of an operator such as $[I - (t/2)\,\nu\nabla^2]$, if it has a regular structure, for later inversion.

2. *Storage.* Semi-implicit methods require a small, fixed number of storage areas for velocity fields, whereas the Krylov method stores many velocity fields in order to span the Krylov subspaces. That is, storage requirements for the Krylov method are dominated by the size $N \times K$ of the V matrices. The size of the Krylov subspace K may, however, be chosen for storage economy, and Fig. 4 shows that CPU time may be traded for storage over some range of K from about 20 down to 5 or less.

3. *CPU time.* If a semi-implicit code were to take full advantage of the regular structure of the vector Laplacian in our Couette–Taylor problem, we estimate that the transient calculation shown in Fig. 2a to b would require from three to five times less CPU time than is required by our Krylov method with $K = 20$. This estimate is based on performance results for other fluid-dynamical codes with which we are familiar and on the problem size N. Note that we are comparing our results to codes which fix the timestep t and which do not incorporate automatic error control. The efficiency with which our time evolution method approaches a steady state, however, greatly surpasses that of semi-implicit schemes because ultimately their stability is determined by the CFL condition, so that the size of the timestep t and therefore the CPU time per model time must remain bounded.

4. *Implementation.* Semi-implicit methods attempt to deal with the stiffness of the Navier–Stokes equations by special handling of the viscous term. Our Krylov method, by contrast, is a *general* method for time evolution of stiff nonlinear equations, and we couple it to elementary fluid dynamical subroutines which compute only explicit actions of the Navier–Stokes right side $F(U)$, its Jacobian $DF(U)$,

and the remainder $G(u)$. Implementation of these elementary subroutines is considerably simpler than implementation of a semi-implicit scheme, and to solve another problem in a different geometry or with a different grid, it would only be necessary to change the elementary fluid dynamical routines and the method of solving the pressure Poisson equation, but not the stiff-equation solver. That is to say that most of Sections 3.1 to 3.3 is very general, and only the subroutines $F(U)$, $DF(U)$, $G(u)$, and ΠU are specific to the Couette–Taylor problem. Separating the implementation tasks in this way has the further advantage that the subroutines $F(U)$ and $DF(U)$ may also be used for steady-state solving and, with minor modification, for linear stability analysis, in conjunction with other general Krylov methods which we will describe in the following sections. The use of general Krylov methods for steady-state and stability calculations, as we will see, allows high accuracy to be achieved with reasonable efficiency and with little additional coding effort.

Gallopoulos and Saad [23] have also proposed a Krylov method for exponential propagation, for which they prove a number of results about convergence and stability. The method has been applied by Saad and Semeraro [54] to the fluid-dynamical driven cavity problem. There are several general remarks we can make:

1. *Calculation of the small-matrix exponential.* Rather than exactly exponentiating tH via explicit diagonalization of H, it is possible to economize by using polynomial or rational-function approximations, as is done in [23]. As previously noted, however, the cost of K Arnoldi steps with $DF(U)$ for our Couette–Taylor problem greatly exceeds the $O(K^3)$ cost of explicit diagonalization of H, and thus we prefer to exactly exponentiate tH. More reliable numerical schemes for computing e^{tH} are given in [25]; however, we have found that computing the matrix exponential via the explicit diagonalization technique is adequate for our purposes. A further motivation for diagonalization is that we may evaluate e^{tH} at different times using the same matrices E and E^{-1} and recomputing only $e^{t\Lambda}$.

2. *Number of Krylov subspaces.* In our evaluation of the integral in (3.25) via (3.30), we use a separate Krylov subspace for each of the vector coefficients g_j. It is possible to reformulate the integral equation and the method of evaluating the integral so as to reduce the number of times that the Arnoldi process must be carried out. Gallapoulos and Saad [23] use the same Krylov subspace at different times τ for the case of a linear system with time-dependent forcing. This multiple use is facilitated by their formulation of the integral equation and by their use of general-purpose quadrature formulas to evaluate the integral. In our method the nonlinear residual $G(u)$ is first fitted to a polynomial in t and we generate Krylov subspaces for each vector coefficient of this polynomial. These Krylov subspaces bear no resemblance to the subspace used to evaluate (3.26), and thus the latter subspace cannot be reused. However, the subspaces generated for each of our vector coefficients g_j are very small, typically of dimension $K_j = 2$ only. Thus evaluation of the integral, which is carried out at each functional iteration step, is very inexpensive compared to the evaluation of (3.26) with $K \approx 20$, which is done only once per timestep.

3. *Parallelism.* On a vector machine, the main advantage to be gained for our problem is in the evaluation of $F(U)$ and $DF(U)$, and in the orthogonalization of the Krylov vectors. The tensor-product basis set (2.1) allows the computation of each spatial derivative (e.g., ∂_r) to be vectorized over the other direction (e.g., z). We do not attempt to take maximum advantage of parallelism for manipulations of the small matrix H, although for some problems this may become significant.

4. *Incompressibility.* In our formulation of the fluid dynamical problem, we enforce incompressibility via a fast direct solution of the consistent pressure Poisson equation (2.5) for each evalution of $F(U)$ and $DF(U)$. In this instance we take advantage of the special structure of the linear system, and thus we do not deal directly with the issue of how incompressibility is to be enforced in situations where direct solution of the Poisson problem is not feasible. In contrast, Saad and Semeraro [54] use an artificial compressibility method in which the flow is not kept exactly divergence-free at all times, but which avoids the need to solve the pressure problem for each evaluation of $F(U)$ and of their Jacobian. Their method is more easily generalizable to nonstandard geometries and grids. We note that other forms of the governing equations, such as the stream function/vorticity formulation, avoid the pressure problem altogether but are not generalizable to three-dimensional flows.

4. STEADY-STATE SOLVING USING A NEWTON–KRYLOV METHOD

The second major problem of computational fluid dynamics is steady-state solving. Here one seeks a velocity field U such that $\partial_t U = F(U) = 0$, or, in a more general context one seeks traveling- or rotating-wave solutions which are steady in some appropriate moving or rotating frame of reference. Steady-state solutions which are stable may be obtained by time evolution, as was done for Taylor vortices in the previous section, although this method is often slow. Unstable steady solutions may also be of interest, and since these cannot be obtained via time evolution one must appeal directly to the steady-state problem $F(U) = 0$.

In this section 4 we will recompute the Taylor vortex solution by a steady-state computation using two variants of Newton's method. Our Krylov–Newton methods use the

same subroutines $F(U)$ and $DF(U)$ that were used by the time evolution method of Section 3. We describe these methods using the Krylov notation and definitions introduced in Section 3.1. Our test case will be the computation of the flow of Fig. 2c using the flow of Fig. 2b as the initial guess. After presenting typical timings we will compare our method to other approaches for the steady-state problem.

4.1. *Description of the Newton–Krylov Method*

Given a current estimate $U^{(m)}$ for a steady state, one step of Newton's method consists of solving

$$DF(U^{(m)})\, u^{(m)} = F(U^{(m)}) \tag{4.1}$$

and updating U via

$$U^{(m+1)} = U^{(m)} - u^{(m)}.$$

Here $F(U)$ and $DF(U)$ are as defined in Eqs. (2.10)–(2.11).

The discrete form of (4.1) is a large linear system in N unknowns with a nonsymmetric operator $DF(U)$. With N large, it is not feasible to directly solve this system, and thus we approximate the solution in a Krylov subspace of dimension K. This idea has been variously called *inexact* [2, 3, 16], *iterative* [55], or *truncated* [6, 43] Newton solving and has been analyzed by a number of authors.

Let us introduce the abbreviations $A^{(m)} \equiv DF(U^{(m)})$ for the Jacobian and $b^{(m)} \equiv F(U^{(m)})$ for the nonlinear residual. The normalized linear residual for an approximate solution $u^{(m)}$ is $\|A^{(m)}u^{(m)} - b^{(m)}\|/\|b^{(m)}\|$. It has been proven [2, 55] that, if (4.1) is solved to fixed relative accuracy c at each Newton step, i.e.,

$$\frac{\|A^{(m)}u^{(m)} - b^{(m)}\|}{\|b^{(m)}\|} = c,$$

then the overall Newton iteration converges linearly at rate c, i.e.,

$$\lim_{m\to\infty} \frac{\|b^{(m+1)}\|}{\|b^{(m)}\|} = c. \tag{4.2}$$

On the other hand, if $A^{(m)}u = b^{(m)}$ is solved to increasing accuracy, in particular, if

$$\frac{\|A^{(m)}u - b^{(m)}\|}{\|b^{(m)}\|^2} = c,$$

then the quadratic convergence of Newton's method is preserved [2, 16]:

$$\lim_{m\to\infty} \frac{\|b^{(m+1)}\|}{\|b^{(m)}\|^2} = c.$$

However, as in our method of exponential propagation, we approximately solve (4.1) in a low- and fixed-dimensional Krylov subspace. The accuracy is therefore low and nearly constant for each Newton step, so that only linear convergence (4.2) is achieved.

Let us now drop the superscript (m) and write $Au = b$ for the linear system (4.1). We employ two well-known Krylov techniques, ORTHORES [61] and GMRES [51], for the approximate solution of this system. These are easily described using the notation we have introduced in Section 3.1. The right-hand side vector b is used to generate the Krylov space, i.e., $v_1 = b/\|b\|$.

Recall from Section 3.1 the definitions (3.6)–(3.10) of V, the $N \times K$ matrix of Krylov vectors, and of H, the $K \times K$ representation of A,

$$H \equiv V^{\mathrm{T}} A V,$$

and the Arnoldi decomposition,

$$AV = VH + we_K^{\mathrm{T}}. \tag{4.3}$$

Just as the exponential e^{tA} can be approximated in the Krylov subspace via

$$e^{tA} \approx V e^{tH} V^{\mathrm{T}},$$

the inverse of A can be approximated via

$$A^{-1} \approx V H^{-1} V^{\mathrm{T}}.$$

In practice, this involves solving for y the $K \times K$ system

$$Hy = V^{\mathrm{T}} b = \|b\|\, e_1, \tag{4.4}$$

followed by the matrix-vector multiplication

$$u \equiv Vy. \tag{4.5}$$

This approximation is called ORTHORES ("orthogonal residual") because the resulting residual is orthogonal to the Krylov subspace. To see this, note that eqs. (4.4) and (4.5), and the fact that b belongs to the Krylov subspace, imply that

$$Au = AVy$$
$$b = VV^{\mathrm{T}} b = VHy,$$

so that the residual satisfies

$$Au - b = (AV - VH)\, y$$
$$= we_K^{\mathrm{T}} y.$$

From this it follows that

$$\|Au-b\| = \|w\|\,|e_K^{\mathrm{T}} y|$$
$$V^{\mathrm{T}}(Au-b) = V^{\mathrm{T}} w e_K^{\mathrm{T}} y = 0,$$

since the Arnoldi process maintains $V^{\mathrm{T}}w = 0$.

GMRES ("generalized minimal residual") is an alternative method which produces an approximate solution the norm of whose residual $\|Au-b\|$ is minimal over all vectors u in the Krylov subspace, that is, all vectors of the form $u = Vy$, for some K-vector y. Minimizing $\|AVy-b\|^2$ for each component of y leads to the least-squares equation

$$(AV)^{\mathrm{T}}(AV)\,y = (AV)^{\mathrm{T}} b \tag{4.6}$$

or

$$(AV)^{\mathrm{T}}(AVy-b) = 0.$$

That is, $Au-b$ is required to be orthogonal to AV rather than to V. The matrices required in Eq. (4.6) are constructed economically using Eq. (4.3) as

$$\begin{aligned}(AV)^{\mathrm{T}} &= (VH + we_K^{\mathrm{T}})^{\mathrm{T}}\\ &= H^{\mathrm{T}}V^{\mathrm{T}} + e_K w^{\mathrm{T}}\\ (AV)^{\mathrm{T}}(AV) &= H^{\mathrm{T}}V^{\mathrm{T}}VH + H^{\mathrm{T}}V^{\mathrm{T}}we_K^{\mathrm{T}}\\ &\quad + e_K w^{\mathrm{T}}VH + e_K w^{\mathrm{T}}we_K^{\mathrm{T}}\\ &= H^{\mathrm{T}}H + \|w\|^2 e_K e_K^{\mathrm{T}},\end{aligned}$$

where we have used the orthogonality of the K Krylov vectors (columns of V) to one another and to w. The right-hand side of (4.6) is

$$\begin{aligned}(AV)^{\mathrm{T}} b &= H^{\mathrm{T}}V^{\mathrm{T}}b + e_K w^T b\\ &= H^{\mathrm{T}}\,\|b\|\,e_1.\end{aligned}$$

Thus, the $K\times K$ system solved by GMRES is

$$(H^{\mathrm{T}}H + \|w\|^2 e_K e_K^{\mathrm{T}})\,y = H^{\mathrm{T}}\,\|b\|\,e_1.$$

For either GMRES or ORTHORES, note that if $\|w\| = 0$ then

$$Au = AVy = VHy = VV^{\mathrm{T}}b = b,$$

so that u will be an exact solution to the linear system, although this is never achieved in practice.

Generally, ORTHORES and GMRES are used as iterative methods: the number of Krylov vectors K is increased until the desired degree of convergence, measured by some norm of the residual $Au-b$, is attained. As K is increased, additional rows and columns are appended to the matrices H and V. However, as in our time-evolution method, we instead choose to work within a Krylov space and fixed size K, solving each of the linear problems $Au = b$ only to a fixed (low) accuracy. Our storage requirements are dominated by the size $N\times K$ of the matrix V, and we cannot afford to solve the linear system to arbitrary accuracy, except by restarting. We assume that it is generally better to take a Newton step rather than to restart, and this is the strategy we have adopted.

4.2. *Taylor Vortex Test Computations*

Using the flow in Fig. 2b as the initial guess, we compute the steady-state Taylor vortex flow; that is, we compute the time-asymptotic flow of Fig. 2c, but via Newton–Krylov iteration rather than by time evolution. Figure 6 shows the convergence of the quasi-Newton iteration for fixed $K = 20$, for both ORTHORES and GMRES. In both cases, $\log \|F(U^{(m)})\|$ decreases linearly with increasing CPU time (which is proportional to m), until a "noise floor" near 10^{-9} which is primarily due to roundoff error in the solution of the pressure Poisson problem. Except for an initial abrupt increase, the norm obtained using GMRES decreases monotonically, while use of ORTHORES leads to oscillations of unknown origin in the nonlinear residual $F(U^{(m)})$. (Such oscillations have also been reported in [5].)

Each choice of K corresponds to a certain level of accuracy for the GMRES or ORTHORES solution at each Newton step and a certain level of efficiency for the overall

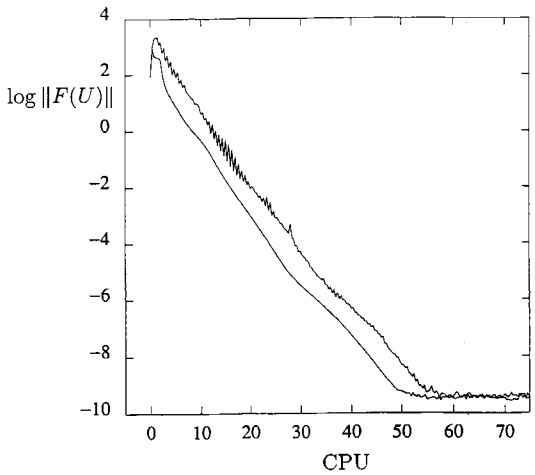

FIG. 6. Evolution of the Krylov–Newton solution of the steady-state problem. The logarithm of the right-hand side $\|F(U)\|$ of Eq. (4.1) is shown as a function of CPU time. The smooth curve results from using GMRES to approximately solve the linear equation; the jagged curve uses ORTHORES. For both methods, the size K of the Krylov space is fixed at $K = 20$ throughout. The linear convergence of the quasi-Newton iteration is evident, as is the presence of a "noise floor" at $\approx 10^{-9}$. The inferior performance of ORTHORES results primarily from error induced in the first few Newton steps.

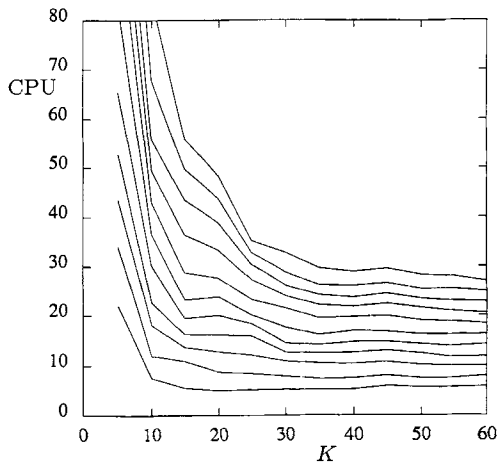

FIG. 7. Efficiency of Krylov–Newton iteration versus the dimension K of the Krylov subspaces used in GMRES. Each curve is a contour of constant $\log \|F(U)\|$; the values are $+1, 0, -1, \ldots, -9$ from bottom to top.

method. Figure 7 shows the efficiency of the Newton-GMRES method as K is varied. We learn that after about $K = 30$ there is little to be gained by adding more dimensions to the Krylov subspace. Regarding solution accuracy at each step, for $K = 20$, for example, we obtain an approximate solution with normalized residual $\|Au - b\|/\|b\| \approx 0.8$ to 0.9, i.e., only a slight improvement over the trivial guess $u = 0$. For $K = 60$, $\|Au - b\|/\|b\| \approx 0.12$ is obtained. As discussed earlier in this section, solving $Au = b$ to fixed accuracy c should lead to linear convergence of the overall Newton iteration at rate c, and this is indeed borne out by our experiments. We believe that effective preconditioners would greatly increase the speed of this convergence by increasing the accuracy of the solution at each iteration.

4.3. *Comparison with Other Methods*

There exist a number of other ways to calculate steady states via Newton's method. Although the linear equation (4.1) is sometimes solved directly by Gaussian elimination, this constrains the spatial resolution and Reynolds number to be extremely low; for most realistic computational fluid dynamics calculations, $DF(U^{(m)})$ is too large to be stored, let alone decomposed.

Equation (4.1) may also be solved iteratively, but it is impossible to achieve reasonable accuracy in a reasonable amount of time. The reason for this is that the large range of eigenvalues of $\nu\nabla^2$ is reproduced in $DF(U^{(m)})$, yielding poorly conditioned matrices for which iterative solution methods, such as conjugate gradient, converge slowly. It is therefore necessary to *precondition* the problem. An obvious candidate for a preconditioner is the inverse of $\nu\nabla^2$, or of the full Stokes operator $\Pi\nu\nabla^2$, and this approach has been used successfully be previous authors, e.g., [3, 5, 60].

Our use of the inexact Newton's method, in which (4.1) is solved iteratively, but only to low accuracy, has been motivated by a desire to avoid viscous operator inversion and to require only explicit evaluation of $F(U^{(m)})$ and $DF(U^{(m)})\,u$. Another important difference between our approach and that of some other authors [2, 3, 6, 54] is that we actually evaluate the action of the exact Jacobian $DF(U^{(m)})\,u$, rather than a finite difference approximation $[F(U^{(m)} + \sigma u) - F(U^{(m)})]/\sigma$.

5. LINEAR STABILITY ANALYSIS VIA AN IMPLICITLY RESTARTED ARNOLDI PROCESS

The third major problem of computational fluid dynamics is linear stability analysis. Given a solution U of the Navier–Stokes equations, one seeks to determine if it is stable with respect to small perturbations u. This leads to an eigenproblem with which the stability of U is determined by examining the real parts of the eigenvalues; the real part of an eigenvalue is the temporal growth rate of the corresponding eigenmode. If one or more eigenvalues have positive real part, U is unstable. In general we need only the *leading* eigenvalue or complex-conjugate pair to make this test. By *leading* we mean an eigenvalue whose real part is greater than or equal to the real part of any other eigenvalue. It may occur that other eigenvalues have real parts close to that of a leading eigenvalue, and these may also be of interest. The remainder of the spectrum, however, corresponds to strongly damped eigenmodes which are not important for the linear stability analysis. If U and u are represented by a large number of unknowns, there will be many more uninteresting strongly damped eigenmodes than interesting leading and nearly leading eigenmodes.

In this section 5 we examine the stability of the Taylor vortex solution U with respect to perturbations corresponding to wavy vortices. We will begin with a description of the fluid dynamical stability problem and the discrete eigenproblem that results from our spatial discretization. We will write this in terms of a modified Jacobian $DF_{mq}(U)$, where the subscripts mq specify the symmetry of the eigenmode we are seeking. The action of this linear operator is computed by an elementary subroutine similar to the Jacobian $DF(U)$ used in the previous sections. We then describe a Krylov method for finding the leading and nearly leading eigenvalues and eigenvectors of this large, nonsymmetric operator. We test the method by computating wavy-vortex eigenmodes, such as Fig. 8, for the linearization about the Taylor vortex field of Fig. 2c. After presenting typical timings, we compare our method to other approaches for large linear stability problems.

5.1. *The Taylor Vortex to Wavy Vortex Eigenproblem*

The fluid dynamical eigenproblem is obtained by linearizing the Navier–Stokes equations around the Taylor vortex

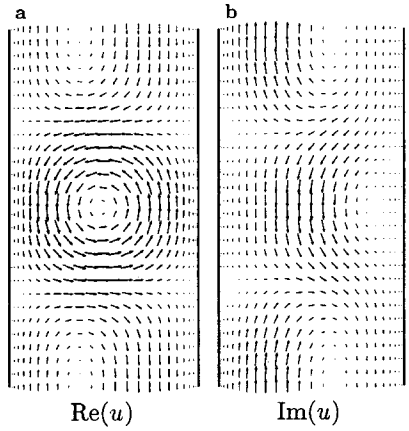

FIG. 8. The real and imaginary parts of $u(r, z)$ of the leading wavy vortex eigenmode for the Taylor vortex flow of Fig. 2c, for azimuthal wavenumber $m = 3$ and Floquet exponent $q = 0$.

solution U. Because U is steady, axisymmetric, and has axial periodicity α, eigenmodes of the linear problem will have the form

$$u(r, z \bmod \alpha)\, e^{im\theta + iqz + \lambda t} + \text{c.c.}, \tag{5.1}$$

where m is an integer azimuthal wavenumber, q is an axial Floquet exponent which we take to be real, and λ is the complex temporal eigenvalue. Eigenmodes of this form are rotating waves with azimuthal wavespeed $|\mathrm{Im}(\lambda)|/m$. Finding the leading eigenmode generally involves a search over all m and q, but one may usually restrict the search to small integers m and small Floquet exponents $q \approx 0$. A related publication [20] examines the role of the Floquet exponent q, especially with regard to finite-length experiments. The present computational problem is to determine for given values of m and q the leading and nearly leading eigenvalues λ and the corresponding eigenmodes u.

Since u in (5.1) is axisymmetric and axially periodic, it may be spatially discretized using the same Chebyshev–Fourier series (2.1) that was used for Taylor vortices, with the proviso that u, and thus the expansion coefficients f_{nj}, are complex. If we rewrite (2.1) for some component f of the total perturbation (5.1) we obtain

$$f(r, \theta, z, t) \approx \sum_{n=0}^{N_r} T_n(x_r(r)) \sum_{j=-N_z/2}^{N_z/2} e^{2\pi ijz/\alpha} \times f_{nj} e^{im\theta + iqz + \lambda t} + \text{c.c.}$$

and we immediately see that when operating on u the azimuthal and axial derivatives must be modified such that $\partial_\theta \to im$ and ∂_z multiplies each f_{nj} by $2\pi ij/\alpha + iq$ and not just $2\pi ij/\alpha$. We introduce the subscript notation ∇_{mq} to indicate this modification. The modified Jacobian is then defined by

$$DF_{mq}(U)\, u = \Pi_{mq}[-((U_C + U)\cdot\nabla_{mq})\, u - (u\cdot\nabla)(U_C + U) + \nu\nabla_{mq}^2 u].$$

The projector Π_{mq} is obtained in the analogous way by noting that the pressure perturbation must also be of the form (5.1) with p in place of u, so that in Eqs. (2.6) and (2.7) we replace ∇ by ∇_{mq} to obtain the explicit definition

$$\Pi_{mq} u \equiv B[u - \nabla_{mq} p], \quad \text{where} \quad \nabla_{mq}\cdot B\nabla_{mq} p = \nabla_{mq}\cdot Bu.$$

The eigenproblem may now be stated as

$$DF_{mq}(U)\, u = \lambda u. \tag{5.2}$$

Figure 8 shows a computed wavy vortex eigenmode. We note that since u is complex, its representation as a vector requires twice the number of real unknowns as the Taylor vortex field U, that is, $(N_r + 1)\times N_z \times 3 \times 2 = 4050$ for $N_r = 24$ and $N_z = 27$. Since the matrix form of $DF_{mq}(U)$ is never constructed, it may be considered to be a complex operator for 2025 complex unknowns or equivalently as a real operator for 4050 real unknowns. In the following section we consider $A = DF_{mq}(U)$ to be a real operator for vectors of length 4050.

5.2. *A Krylov Method for Large Nonsymmetric Eigenproblems*

Suppose that, starting with some initial vector v_1, we have carried out K steps of the Arnoldi process to arrive at the decomposition

$$AV = VH + we_K^{\mathrm{T}} \tag{5.3}$$

as in (3.10). The method we use to compute the leading and nearly leading eigenpairs of $A = DF_{mq}(U)$ is based upon the following considerations:

1. *Approximate* eigenpairs (u_k, λ_k) of A may be obtained from the *exact* eigenpairs (y_k, λ_k) of H by setting $u_k = Vy_k$, $k = 1, \ldots, K$.

2. The matrices V, H and the residual vector w all depend on the initial vector v_1 that generates the Krylov subspace, and thus the approximate eigenpairs will also depend on v_1.

3. The dependence of residual vector w on v_1 is such that $w = 0$ if and only if v_1 is a member of an invariant subspace of dimension K.

4. Using the spectrum of H, we will see that the matrices V, H, and we_K^{T} may be updated iteratively such as to drive v_1 into an invariant subspace spanned by only the leading and nearly leading eigenvectors of A. As a result the approximate eigenpairs converge to exact eigenpairs of A.

The method we use is developed and described fully by Sorensen in [56]. At each iteration we begin with an

Arnoldi decomposition of size K. The small $K \times K$ matrix H is explicitly diagonalized to yield its eigenvalues λ_k, $k = 1, ..., K$. These are sorted in order of increasing real part, keeping complex-conjugate pairs together. The first K_u of them, i.e., those with most negative real part, are identified as unwanted. Typically $K = 20$ and $K_u = 12$. The remaining $K_w = K - K_u$ are wanted, meaning leading or nearly leading. For each unwanted eigenvalue λ_k, $k = 1, ..., K_u$, we apply a shift to the matrices V, H, and we_K^T in a manner analogous to shifted QR iteration, as we will explain in the next few paragraphs. The effect of applying these K_u shifts is to replace the starting vector v_1 by $\psi(A)\, v_1$, where ψ is a filter polynomial of degree K_u with $\psi(\lambda_k) = 0$ for each unwanted λ_k. This filtering is accomplished implicitly, that is, without computing any explicit action of the operator $\psi(A)$. Furthermore, by discarding K_u columns of the updated matrices, we arrive at an Arnoldi decomposition of size K_w, which is exactly the same as that which would have resulted from K_w Arnoldi steps beginning with the vector $\psi(A)\, v_1$, but this is accomplished implicitly without actually restarting the Arnoldi process. The decomposition of size K_w is then restored to size K by simply resuming the Arnoldi process at step $K_w + 1$ and performing an additional K_u steps. The next iteration begins with this new decomposition of size K. The overall method may be described as an implicitly restarted Arnoldi process with implicit polynomial filtering of the starting vector.

To see how and why this works, we need to examine the method of applying shifts. Beginning with $k = 1$, let us suppose that we have performed a QR decomposition of the shifted matrix $H - \lambda_1 I$, for the unwanted eigenvalue λ_1, to obtain Q and R such that

$$H - \lambda_1 I = QR, \tag{5.4}$$

where Q is orthogonal and R is upper triangular, both of size $K \times K$. In essence, our method simply computes updated matrices VQ, $Q^T H Q$, and $we_K^T Q$ which then replace V, H, and we_K^T. We then return to (5.4) for the next unwanted eigenvalue λ_2. The process is repeated for all K_u unwanted eigenvalues, and in the final result (5.3) has been replaced by

$$AV\hat{Q} = V\hat{Q}\hat{Q}^T H\hat{Q} + we_K^T\hat{Q}, \tag{5.5}$$

where $\hat{Q}$ is the product of all the Q's.

The effect of this can be understood by examining a single shift, for example, the first one. Substituting $H = QR + \lambda_1 I$ into (5.3) and rearranging, we obtain

$$(A - \lambda_1 I)\, V = VQR + we_K^T.$$

Equating the first column on both sides then yields

$$(A - \lambda_1 I)\, v_1 = VQe_1 \rho_{11},$$

where VQe_1 is the updated starting vector, i.e., the first column of VQ, and $\rho_{11} = e_1^T R e_1$. Thus by applying *one* shift we have implicitly filtered the starting vector by the polynomial $\rho_{11}^{-1}(A - \lambda_1 I)$ without any explicit computation of the action of this operator. It can also be shown that after all K_u shifts have been applied the starting vector v_1 has been replaced by the implicitly filtered vector $V\hat{Q}e_1 = \psi(A)\, v_1$.

There are several important details that have been left out of this brief description. It is true, but perhaps not obvious, that (5.5) may be written in the form of a legitimate Arnoldi decomposition of size K_w, essentially by discarding K_u columns of $V\hat{Q}$ and $we_K^T\hat{Q}$ and using the upper left $K_w \times K_w$ submatrix of $\hat{Q}^T H\hat{Q}$. Also, in the actual implementation each Q matrix is implicitly computed and applied one Givens transformation at a time. For further information on these issues we refer the reader to [56], which also deals with the issues of maintaining orthogonality of V, elimination of spurious eigenvalues, the handling of numerically small subdiagonal elements of H, numerical stability, and the use of real arithmetic for complex-conjugate pairs of shifts.

At the end of each iteration, we compute the eigenvalue residual $\|(A - \lambda I)\, u\|$ of the leading approximate eigenpair $(u, \lambda) = (Vy, \lambda)$ for u normalized to $\|u\| = 1$. This is done explicitly by computing $u = Vy$ and calling the subroutine $A = DF_{mq}(U)$. In exact arithmetic this residual would be equivalent to the expression $\|w\|\, |e_K^T y|$, as can be seen by multiplying (5.3) on the right by y and using $Vy = u$ and $Hy = y\lambda$ to obtain

$$\begin{aligned} AVy &= VHy + we_K^T y \\ &= Vy\lambda + we_K^T y \end{aligned}$$

so that

$$(A - \lambda I)\, u = we_K^T y. \tag{5.6}$$

This implies that we could avoid the explicit computation of the eigenvalue residual, for example, in stopping tests. Note also that if the residual vector w were zero, then the approximate eigenpairs would all be exact eigenpairs of A.

A proof of the convergence of the implicit restart method described here exists [56] only for the case of a symmetric operator A.

5.3. *Taylor Vortex to Wavy Vortex Test Computations*

In Fig. 9 we show the evolution of the eigenvalue residual norm for the leading approximate eigenvector as a typical calculation progresses. The original starting vector v_1 in this case was created from a sequence of pseudorandom numbers. The final computed leading eigenmode is depicted in Fig. 8. In addition to the straightforward calculation of $\|(A - \lambda I)\, u\|$, the residual norm may be calculated as

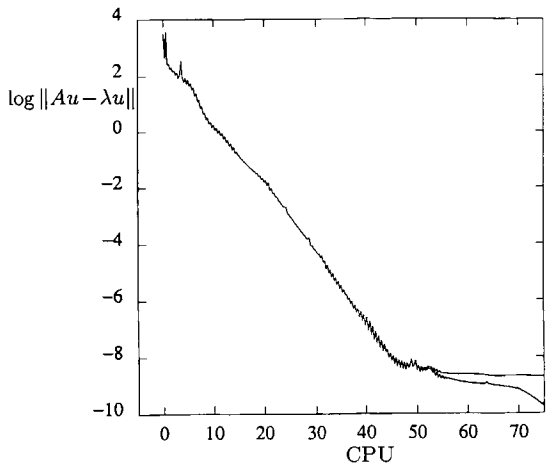

FIG. 9. Convergence history of the leading eigenvalue residual for the implicit polynomial method with $K_w = 8$ and $K_u = 12$. The abscissa shows the expenditure of CPU time. The logarithms of the norms of both $Au - \lambda u$ and of $we_K^T y$ are plotted. Although formally equal and indistinguishable throughout most of the calculation, these eventually differ, with $\|Au - \lambda u\|$ becoming constant while $\|we_K^T y\|$ continues to decrease.

$\|w\| \, |e_K y|$ as in (5.6), and both of these quantities are plotted in Fig. 9. They are equal until a value of approximately 10^{-8} is attained, after which $\|w\| \, |e_K y|$ continues to decrease while $\|(A - \lambda I) u\|$ remains essentially constant. A probable cause for this is roundoff error in the subroutine $DF_{mq}(U)$, especially in the solution of the pressure Poisson problem. This implies that, in practice, acting with the Jacobian $DF_{mq}(U)$ is not exactly equivalent to multiplication by a matrix, so the mathematical equivalence of the two eigenvalue residual calculations carries over to the numerical method only up to this accuracy limit. We note also that the residual norm oscillates in a manner reminiscent of the Krylov–Newton method using ORTHORES (Fig. 6), but again the reason for this is not known.

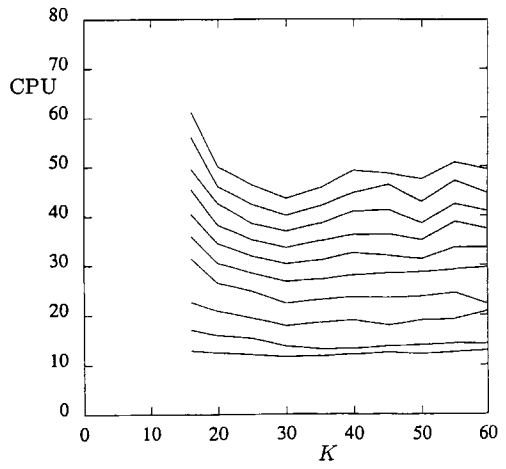

FIG. 10. Efficiency of implicit polynomial method as the total size $K = K_w + K_u$ of the Krylov space is varied. In all cases $K_w = 8$ while K_u ranges from 8 to 52. The curves show contours of constant $\log \|Au - \lambda u\|$ for values $+1, 0, -1, \ldots, -8$ from bottom to top.

Figure 10 shows contours of constant $\|(A - \lambda I) u\|$ as functions of both CPU time and of the total size of the Krylov space, $K = K_w + K_u$, for fixed $K_w = 8$. Recall that wavy vortices are rotating waves and thus the leading and nearly leading eigenmodes are complex-conjugate pairs; specifying $K_w = 8$ allows us to compute four such rotating waves. We see that $K = 30$ is again an optimal value and that choosing $K > 30$ can even increase the cost of calculating an eigenpair to a given accuracy.

5.4. *Comparison with Other Methods*

We now discuss other ways of solving the eigenproblem (5.2). First, the $N \times N$ matrix form of $A \equiv DF(U)$ may be diagonalized by the QR algorithm, yielding all of the eigenpairs directly. Since matrix diagonalization is even more costly than inversion, this is generally ruled out in computational fluid dynamics. The notable exception occurs when the matrix is block diagonal (e.g., [42]) or block banded (e.g., [33]) with small blocks. In this case, only the blocks need be directly diagonalized, and the eigenpairs of the full system is constructed from those of the blocks. Second, variants of the power method for finding dominant eigenvalues (those of largest magnitude) may be used to extract the desired eigenpairs. While the inverse power method might seem a natural choice for calculating eigenvalues of small magnitude, it presents the same difficulties as described in previous sections: in particular, an operator with an eigenvalue near zero is necessarily poorly conditioned, and so iterative solution methods are slow. Instead, a common course [11, 18, 24] is based on the (forwards) power method on the operator e^{tA}, which is accomplished by solving the system $\partial_t u = Au$ by some time-stepping algorithm. The transformation $A \to e^{tA}$ maps the sought-after leading eigenvalues of A to accessible dominant eigenvalues of e^{tA}. The drawback to this method is again intrinsic to the problem: the eigenvalues of A that are near zero are all mapped onto eigenvalues of e^{tA} that are near one (for t small). The ratio between these is close to one, so the power method converges quite slowly.

There are also alternative Krylov methods which restart the Arnoldi process. Variants of this idea were introduced early by Karush in [34]. Cullum and her colleagues have investigated explicit restart methods for the symmetric case [13–15]. Most recently the idea has been explored by Saad in [49] by Chatelin and Ho in [7] and by Chronopoulos in [12] for the nonsymmetric case. In all of these methods, the entire Arnoldi sequence must be restarted. Moreover, in the methods described in [7, 49] an auxiliary calculation involving the explicit calculation of $\psi(A) v_1$ is required. We have described a particular construction of the polynomial

filters. There are many alternatives, including the construction by Saad [49] that is based on Manteuffel's scheme [41], and the variants of this are presented and discussed by Chatelin and Ho in [7].

6. CONCLUSIONS

We have described Krylov methods for each of the three major problems of computational fluid dynamics and we have tested these methods on the specific problem of the Taylor vortex to wavy vortex instability in the Couette–Taylor problem. Our methods rely exclusively on explicit evaluations of the Navier–Stokes right-hand side $F(U)$ and of its Jacobian $DF(U)$, without any inversion of the viscous operator. A major advantage of our overall approach is that the three problems, time evolution, steady-state solving, and linear stability analysis, are solved by conceptually similar methods which use identical or very similar elementary fluid dynamical subroutines.

Because no matrix form of the Jacobian $DF(U)$ is needed, the total number of unknowns N is not constrained by the N^2 storage cost of such a matrix. In practice this means that spatial discretization errors can be reduced by using a finer grid. Furthermore, since the elementary subroutines needed for our time evolution method are the same or very similar to those needed for steady-state and linear stability calculations, the latter two problems are solved essentially with no further implementation effort. There is thus nothing to be saved by using time evolution, as is often done, to study asymptotic steady states or the linear behavior of small perturbations. One can just as easily solve the steady-state and linear stability problems by Krylov methods tailored to those calculations, thus avoiding any temporal discretization error.

We do not claim, however, that these methods are the most efficient ways to solve these problems. We have already noted that semi-implicit methods may well outperform our time evolution method by taking advantage of the special structure of the vector Laplacian, if this is possible. The major exception to this is upon close approach to a steady state, where our Krylov method speeds up dramatically. For Krylov–Newton steady-state solving, effective preconditioning of the linear system can greatly improve the accuracy achieved at each Newton step, and thus also the convergence rate, as shown, for example, in [3, 5, 60].

However, Krylov methods based exclusively on explicit actions of the Navier–Stokes right-hand side $F(U)$ and its Jacobian $DF(U)$ are easily implemented and reasonably efficient. While we have focused specifically on the Couette–Taylor problem, Krylov methods could well be used in many other situations. Spatial discretization schemes including finite difference, finite element, spectral, and spectral element are compatible with these methods. In addition, the extension of the methods to more complicated governing equations, including, for example, thermal, chemical, or electromagnetic effects, is relatively straightforward since the elementary fluid dynamical subroutines would simply compute explicitly the action of each new term. Due to this flexibility we expect that Krylov methods will find increasing use for a wide variety of practical flow problems.

ACKNOWLEDGMENTS

The authors are supported in part by the Office of Naval Research Nonlinear Dynamics Program (WSE), by NSF Grant DMS-8901767 (LST), by the Welch Foundation (RAF), and by DOE contract DE-FGof-91ER25103 and NSF cooperative agreement CCR-9120008 (DCS). Computational resources for this work were provided by the University of Texas System Center for High Performance Computing. We thank Katie Coughlin, Joel Franklin, Philip Gresho, Wayne Joubert, Philip Marcus, Jonathan Poritz, Nick Trefethen, Karen Uhlenbeck, and Xiadong Zhang for valuable comments and suggestions.

REFERENCES

1. W. E. Arnoldi, *Quart. Appl. Math.* **9**, 17 (1951).
2. P. N. Brown, *SIAM J. Numer. Anal.* **24**, 417 (1987).
3. P. N. Brown and Y. Saad, *SIAM J. Sci. Stat. Comput.* **11**, 450 (1990).
4. C. Canuto, M. Y. Hussaini, A. Quateroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics* (Springer-Verlag, New York, 1988).
5. G. F. Carey, K. C. Wang, and W. D. Joubert, *Int. J. Numer. Methods Fluids* **9**, 127 (1989).
6. T. F. Chan and K. R. Jackson, *SIAM J. Sci. Stat. Comput.* **5**, 533 (1984).
7. F. Chatelin and D. Ho, *Math. Modeling Numer. Anal.* **24**, 53 (1990).
8. A. Chorin, *Math. Comput.* **22**, 745 (1968).
9. A. Chorin, *Math. Comput.* **23**, 341 (1968).
10. A. Chorin, *Stud. Numer. Anal.* **2**, 64 (1968).
11. K. N. Christodoulou and L. E. Scriven, *J. Sci. Comput.* **3**, 355 (1988).
12. A. T. Chronopoulos, Dept. Computer Science Report TR 90-15, University of Minnesota, Minneapolis, MN., 1989.
13. J. Cullum and W. E. Donath, in *Proceedings, 1974 IEEE Conference on Decision and Control* (IEEE Press, New York, 1974), p. 505.
14. J. Cullum, *BIT* **18**, 265 (1978).
15. J. Cullum and R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations* (Birkhauser, Boston, 1985).
16. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *SIAM J. Numer. Anal.* **19**, 400 (1982).
17. R. C. Diprima and H. L. Swinney, in *Hydrodynamic Instabilities and the Transition to Turbulence*, edited by H. L. Swinney and J. P. Gollub (Springer-Verlag, New York, 1981).
18. W. S. Edwards, in *Instability and Transition*, edited by M. Y. Hussaini and R. G. Voigt (Springer-Verlag, New York, 1990).
19. W. S. Edwards, Ph.D. thesis, University of Texas, Austin, 1991.
20. W. S. Edwards, S. R. Beane, and S. Varma, *Phys. Fluids A* **3**, 1510 (1991).
21. J. Franklin, private communication.
22. R. A. Friesner, L. S. Tuckerman, B. C. Dornblaser, and T. V. Russo, *J. Sci. Comput.* **4**, 327 (1989).
23. E. Gallopoulos and Y. Saad, *SIAM J. Scient. Stat. Comput.* **13**, 1236 (1992).
24. I. Goldhirsch, S. A. Orszag, and B. K. Maulik, *J. Sci. Comput.* **2**, 33 (1987).

25. G. H. Golub and C. F. Van Loan, *Matrix Computations* (The Johns Hopkins, Baltimore, 1983).
26. D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications* (SIAM, Philadelphia, 1977).
27. P. M. Gresho, *Int. J. Numer. Methods Fluids* **11**, 587 (1990).
28. P. M. Gresho, *Int. J. Numer. Methods Fluids* **11**, 621 (1990).
29. P. M. Gresho, in *Annu. Rev. Fluid Mech.* **23** (1991).
30. P. M. Gresho, S. T. Chan, R. L. Lee, and C. D. Upson, *Int. J. Numer. Methods Fluids* **4**, 557 (1984).
31. P. M. Gresho and R. L. Sani, *Int. J. Numer. Methods Fluids* **7**, 111 (1987).
32. C. A. Jones, *J. Fluid Mech.* **102**, 249 (1981).
33. C. A. Jones, *J. Comput. Phys.* **61**, 321 (1985).
34. W. Karush, *Pacific J. Math.* **1**, 233 (1951).
35. L. Kleiser and U. Schumann, in *Proceedings, Third GAMM Conference on Numerical Methods in Fluid Mechanics*, edited by E. H. Hirschel (Vieweg, Braunschweig, 1980), p. 165.
36. E. R. Krueger, A. Gross, and R. C. DiPrima, *J. Fluid Mech.* **24**, 521 (1966).
37. H. C. Ku, T. D. Taylor, and R. S. Hirsh, *Comput. Fluids* **15**, 195 (1987).
38. C. C. Lanczos, *J. Res. Nat. Bur. Standards* **49**, 33 (1952).
39. C. Leforestier, R. Bisseling, C. Cerjan, M. Feit, R. Friesner, A. Guldberg, A. Hammerich, G. Jolicard, W. Karrlein, H. D. Meyer, N. Lipkin, O. Roncero, and R. Kosloff, *J. Comput. Phys.* **94**, 59 (1991).
40. W. F. Langford, R. Tagg, E. J. Kostelich, H. L. Swinney, and M. Golubitsky, *Phys. Fluids* **31**, 776 (1988).
41. T. A. Manteuffel, *Numer. Math.* **31**, 183 (1978).
42. P. S. Marcus, *J. Fluid Mech.* **146**, 45 (1984).
43. S. G. Nash, *SIAM J. Scient. Stat. Comput.* **6**, 599 (1985).
44. S. A. Orszag, *J. Fluid Mech.* **50**, 689 (1971).
45. T. Park and J. C. Light, *J. Chem. Phys.* **85**, 5870 (1986).
46. B. N. Parlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, NJ, 1980).
47. Y. Saad, *Linear Algebra Appl.* **34**, 269 (1980).
48. Y. Saad, *Math. Comput.* **37**, 105 (1981).
49. Y. Saad, *Math. Comput.* **42**, 567 (1984).
50. Y. Saad, *SIAM J. Sci. Stat. Comput.* **5**, 203 (1984).
51. Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7**, 856 (1986).
52. Y. Saad, *Comput. Phys. Commun.* **53**, 71 (1989).
53. Y. Saad, *SIAM J. Numer. Anal.* **20**, 209 (1992).
54. Y. Saad and D. Semeraro, *AIAA Pap.* 91-1567.
55. A. H. Sherman, *SIAM J. Numer. Anal.* **15**, 755 (1978).
56. D. Sorensen, *SIAM J. Matrix Anal. Appl.* **13**, 357 (1992).
57. D. Sorensen, in *Proceedings, Cornell MSI Workshop on Large-Scale Numerical Optimization*, edited by T. Coleman and Y. Li (SIAM, Philadelphia, 1991).
58. G. I. Taylor, *Philos. Trans. R. Soc. A* **223**, 289 (1923).
59. L. S. Tuckerman, *J. Comput. Phys.* **80**, 403 (1989).
60. L. S. Tuckerman, in *Proceedings, Eleventh Int'l. Conf. on Numerical Methods in Fluid Dynamics*, edited by D. L. Dwoyer, M. Y. Hussaini, and R. G. Voigt (Springer-Verlag, New York/Berlin, 1989).
61. D. M. Young and K. C. Jea, *Linear Algebra Appl.* **34**, 159 (1980).