DISSERTATION

TIME-STEPPER BASED NUMERICAL BIFURCATION ANALYSIS: AN APPLICATION TO THE TAYLOR-COUETTE PROBLEM

Submitted by

Beau Grande

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2005

COLORADO STATE UNIVERSITY

March 30, 2005

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY BEAU GRANDE ENTITLED "TIME-STEPPER BASED NUMERICAL BIFURCATION ANALYSIS: AN APPLICATION TO THE TAYLOR-COUETTE PROBLEM" BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Michael Kirby

Michael Kirby

David Dandy

David Dandy

Adviser: James Thomas

Co-Adviser: Simon Tavener

Department Head: Simon Tavener

ABSTRACT OF DISSERTATION

TIME-STEPPER BASED NUMERICAL BIFURCATION ANALYSIS: AN APPLICATION TO THE TAYLOR-COUETTE PROBLEM

We present a numerical bifurcation analysis of the Taylor-Couette problem which utilizes a pre-existing time evolution code as the core computational engine. Such time evolution codes, or *time-steppers*, suffer from the drawback that bifurcation-theoretic results cannot easily be obtained from them, if at all. By incorporating relatively minor changes to the underlying time-stepper code, we have developed a computational structure around the existing time-stepper that enables us to perform these bifurcation-theoretic tasks. The cylinder geometry studied has radius ratio 0.615 and aspect ratio 2.4. For a fixed inner Reynolds number $R_i = 300$, three distinct solution branches are analyzed for the range of outer Reynolds number $-320 \leq R_o \leq 0$. These branches exhibit a wide variety of interesting points, including Hopf bifurcation points, symmetry-breaking pitchfork bifurcation points, turning points, and a torus bifurcation point. Unstable steady and time-periodic solutions are also computed.

Beau Grande
Department of Mathematics
Colorado State University
Fort Collins, CO 80523
Spring 2005

ACKNOWLEDGEMENTS

I would like to thank my advisers, James Thomas and Simon Tavener, whose near-infinite patience and keen mathematical insights helped make this dissertation possible. I also thank my parents, Tom and Debbie, for their continued support (both emotional and financial) as I pursued my graduate career. Who knew two years could last so long? A huge thank you to my wife, Kendra. You are my inspiration, and I couldn't have done this without your support. I love you. Finally, the greatest thanks I have goes to God, in whom I live and move and have my being. Without his help, guidance, wisdom, and patience, this dissertation would not have been possible. All glory goes to him.

For Kendra

Contents

Chapter 1

Introduction

Consider a parameter-dependent nonlinear dynamical system described by a system of ordinary differential equations:

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda), \tag{1.1}$$

where $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{f} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$. For the particular application we have in mind, this system results from the spatial discretization of a system of time-dependent partial differential equations, but, in general, this need not be the case. Either way, it is desirable to understand the behavior of this dynamical system; for instance, to be able to compute equilibrium solutions of (1.1), to determine the linear stability of those solutions, and to find bifurcation points where the stability and qualitative nature of those solutions change as the parameter λ is varied.

There are basically two techniques used to accomplish this [109]. The first is to perform direct temporal simulation, viewing the dynamical system as an initial value problem. In this case, an initial condition, boundary conditions, and time horizon are specified. The equations are then integrated by some suitable method over the time horizon to get the state of the system at the desired time. Such a computational routine is often called a "time-stepper" in the literature. Many such simulation codes, after discretization of the temporal derivative, can be idealized as an iteration of the form

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n, \lambda), \tag{1.2}$$

where n counts the time steps. This is essentially a Picard-type iteration. Stable equilibrium solutions, those where $\mathbf{f} = \mathbf{0}$, are obtained by iterating (1.2) until the norm of the difference of successive iterates falls below a user-defined tolerance.

Much information about the dynamical system can be obtained by direct simulation. However, as a bifurcation tool, simulation does have some drawbacks. Near bifurcation points, the rate of convergence of the time-stepper algorithm slows down arbitrarily,[1] causing the expense of computing a solution to grow significantly. As a result, information about solution bifurcations is usually limited to simply observing the change in the solution; the location of any bifurcations, in terms of the parameter λ, is difficult to establish with accuracy. Furthermore, the computation of unstable solutions is impossible with a time-stepper routine, since (1.2) cannot converge to an unstable solution, whereas knowledge of unstable solutions is often desired to understand more fully the system's behavior. These limitations of the time-stepper are due to the fact that bifurcations, the rate of convergence, and the stability of solutions are connected to the eigenvalues of the Jacobian matrix $\mathbf{F_u}(\mathbf{u}, \lambda)$, which are not typically available in a time-stepper code. We explain this connection more fully in Chapter 2.

An alternative to direct simulation is to use numerical bifurcation techniques. Information about equilibrium solutions, unstable solutions, and bifurcations are obtained by augmenting the differential equations describing the dynamical system with equations that describe the particular behavior being sought. For example, a popular method used to follow a solution branch around a turning point is the (pseudo)arclength continuation method of H. B. Keller [73]. This method introduces a new parameter into the system via an extra equation describing the arclength parameter or an approximation of it. Another example is the introduction of test functions to monitor a particular behavior of the system. These functions involve the Jacobian

---

[1]Here, "arbitrarily" is to be interpreted in the sense of limits: the rate of convergence can be made as slow as one pleases by taking a point sufficiently close to the bifurcation point.

matrix of the right-hand side of (1.1). Examples are functions that detect a change in sign of the determinant of the Jacobian matrix or monitor its leading eigenvalues. These test functions can be incorporated as additional equations to form extended systems or by forming a so-called bordered system that is no longer singular at the point of interest.

Extended systems for computing turning points have been proposed by Moore and Spence [102] and for computing Hopf points by Griewank and Reddien [60], Roose and Hlavaček [110] (whose work was later extended by Werner and Janovsky [144]), and by Werner [143]. Multiple Hopf points are treated by Govaerts, Guckenheimer, and Khibnik [57] and symmetry-breaking bifurcation points by Werner and Spence [145]. For multi-parameter systems, where $\lambda \in \mathbb{R}^m$ for some $m > 1$, see the work of Jepson and Spence [70], [126], [69], [71] and Jepson, Spence, and Cliffe [72]. Recently, systems for computing periodic solution bifurcations have been proposed by Doedel, Govaerts, and Kuznetsov [44]. The review articles [15] and [146] also give an overview of these techniques, as does the book by Govaerts [58].

Many software packages have been written to implement these standard numerical bifurcation techniques to compute equilibrium solutions, solution branches, bifurcation points, and much more. One of the most versatile and widely used bifurcation software packages is AUTO, first developed at Concordia University by E. Doedel in 1980. The current version is AUTO97 [43], although a newer version is in beta testing. Doedel, Keller, and Kernévez [41, 42] give complete descriptions of the algorithms used in AUTO. When combined with the HOMCONT package [22], AUTO can also compute and follow homoclinic orbits. The package XPPAUT [52] can be used to simulate systems of ODEs and incorporates AUTO algorithms for bifurcation analysis. Another more widely used software package for analyzing dynamical systems is DSTOOL (Dynamical System Toolkit), which was originally developed at Cornell University [9]. The package LOCBIF (for Local Bifurcation analyzer) is quite similar to AUTO in capability. Its algorithms are based on the work of A. Khibnik

et al. [76]. More recently, a package called CONTENT has been developed by Yu. A. Kuznetsov and V. V. Levitin [80], and improves upon AUTO and LOCBIF in several ways. For example, it can compute double Hopf points, which the others cannot do. Finally, a MATLAB version of CONTENT has recently been developed, called MATCONT [38]. Although the numerical algorithms in MATCONT are the same as those in CONTENT, MATCONT has been entirely redesigned to take advantage of MATLAB's built-in features, such as its ODE suite for time integration and its Symbolic Toolbox for computing derivatives. Other software packages that incorporate these bifurcation techniques include the software package LOCA (Library of Continuation Algorithms), developed at Sandia National Laboratories [114], and the finite element code ENTWIFE [28]. The algorithms used in these latter two packages are amenable to working with systems of very large dimension, so even problems arising from the spatial discretization of PDEs can be solved and analyzed by them.

Even though numerical bifurcation software has become more sophisticated and computing power has continued to increase, there are many situations where considerable effort and time have been put into developing an accurate and efficient time-stepper routine for a given system. Therefore, it would be beneficial to augment this legacy code, without rewriting it, to be able to compute some of the system behaviors not typically accessible by a time-stepper code (such as unstable solutions and more precise bifurcation locations). In other words, we would like to adapt an existing code to perform bifurcation analysis like that given by numerical bifurcation techniques. The so-called time-stepper based numerical bifurcation theory attempts to do just that. By cleverly "wrapping" a computational structure around the existing time-stepper program, essentially treating the existing time-stepper code as a black-box, the goal is to produce a routine that gives many of the results of numerical bifurcation theory while utilizing the existing time-stepper.

Some progress has already been made in this direction. Tuckerman and Barkley [139] have developed methods for transforming an existing time-stepper code into a

bifurcation tool. They consider a dynamical system written in the form

$$\frac{d\mathbf{u}}{dt} = N(\mathbf{u}) + L\mathbf{u},$$

where L and N represent the linear and nonlinear parts, respectively, of the right-hand side. They assume that a time-stepper code has been formed by applying a first-order finite difference to the time derivative and by utilizing an implicit/explicit scheme for the right-hand side, where N is evaluated at time step n and the linear term L is evaluated at time step $n+1$. The time-stepper therefore is of the form

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t[N(\mathbf{u}^n) + L\mathbf{u}^{n+1}].$$

Time-dependent behavior can be analyzed directly through this time-stepper. It could also be used to calculate equilibrium solutions by iterating the code many times. However, Tuckerman and Barkley suggest using Newton's method for this rather than iteration. Newton's method is applied to solving $N(\mathbf{u}) + L\mathbf{u} = 0$, where the time-stepper and its linearization are used to solve the resulting matrix equation for each Newton iteration. The advantage of using Newton's method over time integration is that unstable equilibrium solutions can be computed. When augmented with Krylov subspace methods, one can use this to calculate dangerous eigenvalues and hence gain information about stability. This method has been successfully applied to the study of spherical Couette flow by Mamun and Tuckerman [92] and to Marangoni convection in mixed fluids by Bergeon, Henry, BenHadid, and Tuckerman [14]. We note that small changes to the original time-stepper code are needed to compute its linearization, and so the method does not treat the time-stepper as a true black-box code. Furthermore, because of the assumed structure of the time-stepper, these techniques cannot be utilized with an arbitrary time-stepper code.

Finally, we mention the work of Koronaki, Boudouvis, and Kevrekidis [77], who have written a program that enables existing time-stepper codes that model tubular reactors to perform bifurcation studies. They use a method called the recursive

projection method (RPM), originally developed by Shroff and Keller [121]. We will discuss this method in detail in Chapter 3. On a related front, Kevrekidis and others ([54, 75, 79, 91, 101, 112, 134]), have used microscopic-level codes written to describe chemical processes to perform a "coarse" macroscopic bifurcation analysis when the governing equations (1.1) are not explicitly known. Their methods are based on the RPM as well.

This dissertation describes a wrap-around code for performing a time-stepper based numerical bifurcation analysis of the Taylor-Couette problem. Many temporal simulation codes for the Taylor-Couette problem have been written. We will focus on that written by Adair [2]. We have successfully written a wrap-around code for his program based on the Newton-Picard method of Lust et al. [89], which we have used to perform a bifurcation analysis of the Taylor-Couette problem.

There are connections between bifurcations of solutions of (1.1) and the convergence of the time-stepper iteration (1.2), but these connections can be subtle. In Chapter 2, we give details of these connections, and in particular, we show how eigenvalue information about the time-stepper can be used for a bifurcation study of the dynamical system. Chapter 3 outlines the numerical methods used to extract this information and shows how they can be used in a computational structure around the time-stepper. The Taylor-Couette problem has been analyzed extensively because it is a superb example for studying hydrodynamic stability and change. We describe the problem in more detail in Chapter 4, along with the details of the time-stepper code used by Adair. Details of implementing our computational structure around Adair's time-stepper code are given in Chapter 5, as are the tests performed to verify the wrap-around code. Chapter 6 describes the numerical experiments conducted with the wrap-around code. Finally, Chapter 7 gives conclusions of our work and points out areas for future work. The Appendices give technical details of some aspects of the program.

Chapter 2

Time-Steppers and Bifurcations

In this chapter, we present some of the basic theory behind linear stability and bifurcation analysis of dynamical systems given by systems of time-dependent ODEs. We then look at the mathematics of time-steppers, placing them within the context of an iterative mapping. We show how information from the time-stepper can be used to determine solution bifurcations of the original dynamical system. In addition, we explain why time-steppers are not well suited for bifurcation analysis. Finally, we describe some of the desirable properties that must be possessed by the numerical methods used to adapt a time-stepper code into a bifurcation tool.

First, we consider the case of computing equilibrium solutions. The case of time-periodic solutions is presented in Section 2.2, since this case is actually easier to deal with than the equilibrium case.

2.1 Equilibrium Solutions

2.1.1 Linear Stability and Bifurcation Theory

In this section, we make rigorous the notions of stable and unstable equilibrium solutions and state the main theorem concerning stability of these solutions. Consider the parameter-dependent dynamical system

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda), \tag{2.1}$$

where for each t, $\mathbf{u}(t) \in \mathbb{R}^N$ and $\mathbf{f} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$. We assume that $\mathbf{f}$ is autonomous, that is, it does not explicitly depend on time t. Recall that an equilibrium solution of (2.1) is a solution $(\mathbf{u}_0, \lambda_0)$ for which $\mathbf{f}(\mathbf{u}_0, \lambda_0) = \mathbf{0}$. We give the following definition of stability from [61].

Definition 2.1. *Let $\mathbf{u}_0$ be an equilibrium solution of (2.1). Then, $\mathbf{u}_0$ is called **stable** if for every $\epsilon > 0$ there exists a $\delta > 0$ such that*

$$\|\mathbf{u}(t) - \mathbf{u}_0\| < \epsilon$$

*for all $t \geq 0$ and for all solutions $\mathbf{u}(t)$ of (2.1) satisfying $\|\mathbf{u}(0) - \mathbf{u}_0\| < \delta$. The equilibrium solution $\mathbf{u}_0$ is said to be **unstable** if it is not stable. The equilibrium solution $\mathbf{u}_0$ is said to be **asymptotically stable** if it is stable and, in addition, there exists an $r > 0$ such that $\|\mathbf{u}(t) - \mathbf{u}_0\| \to 0$ as $t \to \infty$ for all solutions $\mathbf{u}(t)$ of (2.1) satisfying $\|\mathbf{u}(0) - \mathbf{u}_0\| < r$.*

The main result which governs stability of equilibrium solutions is given in the following theorem, which is an amalgam of Theorems 9.5 and 9.7 of [61].

Theorem 2.2. *If all eigenvalues of the Jacobian matrix $\mathbf{f}_\mathbf{u}(\mathbf{u}_0, \lambda_0)$ have negative real parts, then the equilibrium solution $\mathbf{u}_0$ of (2.1) is asymptotically stable. If at least one eigenvalue of $\mathbf{f}_\mathbf{u}(\mathbf{u}_0, \lambda_0)$ has positive real part, then $\mathbf{u}_0$ is unstable.*

This theorem shows how an equilibrium solution $\mathbf{u}_0$ can lose stability at a bifurcation point as the parameter λ is varied. Stability is lost when at least one eigenvalue of the Jacobian matrix $\mathbf{f}_\mathbf{u}$ crosses the imaginary axis in the complex plane. We will consider two ways in which this can happen: a real eigenvalue crosses the imaginary axis through 0, or a complex conjugate pair of eigenvalues crosses the imaginary axis through $\pm i\omega$ with $\omega \neq 0$. We will not address the higher level singularities that result when, for example, two pairs of complex conjugate eigenvalues cross the imaginary axis simultaneously. We would need to vary two parameters to find these singularities

generically, but here we are considering a dynamical system dependent on only one parameter.

First, consider the case when a real eigenvalue crosses the imaginary axis through 0 as λ is varied. Generically, this indicates that a turning point on the solution branch has been reached. However, if the solution possesses some sort of invariance property, such as $\mathbb{Z}_2$-symmetry, then this situation can also indicate the presence of a symmetry-breaking bifurcation point. As we see in Chapter 6, the Taylor-Couette problem has many solutions which have $\mathbb{Z}_2$-symmetry, so the possibility of finding symmetry-breaking bifurcation points is very real.

Now consider the case where a complex conjugate pair of eigenvalues crosses the imaginary axis through $\pm i\omega$, $\omega \neq 0$. This corresponds to a Hopf bifurcation, whereby the equilibrium solution loses stability to a branch of time-periodic solutions. The theory of Hopf bifurcations also gives an estimate of the initial period T on this time-periodic branch [129], which is

$$T \approx \frac{2\pi}{\omega}. \tag{2.2}$$

2.1.2 Mathematics of Time-Steppers

In this section, we explore some of the mathematical properties of a time-stepper code by viewing it as a mapping. We also discuss how stability of equilibrium solutions of (2.1) are connected to the convergence properties of this time-stepper map.

In general, the time-stepper code is obtained by a suitable discretization of the temporal derivative $d\mathbf{u}/dt$. After imposing this discretization, we can view the time-stepper as an iterative mapping of the form

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n, \lambda), \quad \mathbf{F} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N, \tag{2.3}$$

where the variable n counts the time step. Equilibrium solutions $\mathbf{u}^*$ of (2.1) can be approximated by iterating (2.3) until the norm $\|\mathbf{u}^{n+1} - \mathbf{u}^n\|$ of successive iterates

falls below a user-defined tolerance. Then, $\mathbf{u}^*$ is a fixed point of $\mathbf{F}$, provided $\mathbf{F}$ is continuous: if the sequence $\{\mathbf{u}^n\}$ converges to $\mathbf{u}^*$, then we have

$$\mathbf{u}^* = \lim_{n\to\infty} \mathbf{u}^{n+1} = \lim_{n\to\infty} \mathbf{F}(\mathbf{u}^n) = \mathbf{F}\left(\lim_{n\to\infty} \mathbf{u}^n\right) = \mathbf{F}(\mathbf{u}^*).$$

Note that, in a slight abuse of notation, we let $\mathbf{u}^*$ denote both the fixed point of $\mathbf{F}$ and the equilibrium solution of (2.1).

The convergence of iteration (2.3) is intimately tied to the eigenvalues of the Jacobian matrix $\mathbf{F}_\mathbf{u}(\mathbf{u}^*)$. Indeed, if the spectral radius of the Jacobian matrix is strictly less than one, then the iteration will converge (locally) to a fixed point $\mathbf{u}^*$. This is a consequence of the following more general result:

Theorem 2.3 (adapted from [78], page 21). *Let $\mathbf{F}$ be an operator from a Banach space into itself and let $\mathbf{u}^*$ be a fixed point of $\mathbf{F}$. If $\mathbf{F}$ is Fréchet differentiable at $\mathbf{u}^*$ and if the spectral radius $\rho(\mathbf{F}_\mathbf{u}(\mathbf{u}^*)) < 1$, then the sequence $\{\mathbf{u}^n\}$ defined by*

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n), \quad n = 0, 1, \ldots,$$

converges to $\mathbf{u}^*$, provided that $\mathbf{u}^0$ is sufficiently close to $\mathbf{u}^*$. Moreover, for every $\epsilon > 0$, there exists a constant C, depending only on $\mathbf{u}^0$ and ϵ but not on n, such that

$$\|\mathbf{u}^n - \mathbf{u}^*\| \leq C\left(\rho(\mathbf{F}_\mathbf{u}(\mathbf{u}^*)) + \epsilon\right)^n. \tag{2.4}$$

Conversely, if the spectral radius $\rho(\mathbf{F}_\mathbf{u}(\mathbf{u}^*)) \geq 1$, then the sequence $\{\mathbf{u}^n\}$ defined above does not converge to $\mathbf{u}^*$.

In the case that the spectral radius $\rho(\mathbf{F}_\mathbf{u}(\mathbf{u}^*)) \geq 1$, the iteration (2.3) may either diverge or converge to a different fixed point of $\mathbf{F}$. Note that this theorem also gives a rate of convergence. We will return to this in Section 2.1.3.

Now let us consider the specific time discretization used in the Taylor-Couette time-stepper code of interest to us. We discuss this time-stepper in more detail in Chapter 4, but for now all we need to know is that it uses a forward (explicit) Euler

scheme to discretize the temporal derivative. For fixed $\Delta t > 0$, denote the solution $\mathbf{u}$ at time $n\Delta t$ by $\mathbf{u}^n, n = 0, 1, \ldots$. The forward Euler scheme approximates the temporal derivative by

$$\frac{d\mathbf{u}}{dt} \approx \frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t}.$$

Then, equation (2.1) can be approximated by this explicit scheme as

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{f}(\mathbf{u}^n, \lambda) \equiv \mathbf{F}(\mathbf{u}^n, \lambda), \tag{2.5}$$

which gives the particular form of the mapping $\mathbf{F}$.

Since all that is available is the time-stepper code, we need to establish a connection between bifurcation of equilibrium solutions of (2.1) and convergence properties of the iteration (2.5). From Theorems 2.2 and 2.3, we know that the eigenvalues of both Jacobian matrices $\mathbf{f}_\mathbf{u}$ and $\mathbf{F}_\mathbf{u}$ play critical roles. To see how they are related, we note that iteration (2.5) finds equilibrium solutions of (2.1) by computing fixed points of the map $\mathbf{F}(\mathbf{u}, \lambda) = \mathbf{u} + \Delta t\, \mathbf{f}(\mathbf{u}, \lambda)$. Differentiating this equation with respect to $\mathbf{u}$ yields

$$\mathbf{F}_\mathbf{u} = I + \Delta t\, \mathbf{f}_\mathbf{u}.$$

Denote the eigenvalues of $\mathbf{f}_\mathbf{u}$ by $\{\alpha_k\}_{k=1}^N$ and the corresponding eigenvalues of $\mathbf{F}_\mathbf{u}$ by $\{\mu_k\}_{k=1}^N$. Then, the eigenvalues of the two Jacobian matrices are related by

$$\mu_k = 1 + \Delta t \alpha_k. \tag{2.6}$$

We now state our main result which connects the location of eigenvalues of the Jacobian matrix $\mathbf{F}_\mathbf{u}$ with stability of equilibrium solutions of (2.1).

Proposition 2.4. *With the above cumulative notation:*

(a) Iteration (2.5) *is convergent if and only if*

$$\Re e(\alpha_k) < -\frac{\Delta t}{2}|\alpha_k|^2 \tag{2.7}$$

for all $k = 1, \ldots, N$. *In particular, if iteration* (2.5) *is convergent with fixed point* $\mathbf{u}^*$, *then the equilibrium solution* $\mathbf{u}^*$ *of* (2.1) *is stable.*

(b) If at least one eigenvalue μ of the Jacobian matrix $\mathbf{F_u}(\mathbf{u}^*)$ has real part $\Re e(\mu) > 1$, then the equilibrium solution $\mathbf{u}^*$ of (2.1) is unstable. Moreover, the equilibrium solution $\mathbf{u}^*$ loses stability if and only if an eigenvalue μ of $\mathbf{F_u}$ crosses the line $\Re e(z) = 1$ in the complex plane.

PROOF: (a) By Theorem 2.3, iteration (2.5) is convergent if and only if $|\mu_k| < 1$ for all k. For each eigenvalue α_k of $\mathbf{f_u}(\mathbf{u}^*)$, write $\alpha_k = a_k + b_k i$, where $a_k, b_k \in \mathbb{R}$. For each $k = 1, \ldots, N$, we have

$$\begin{aligned}
|\mu_k| < 1 &\iff |\mu_k|^2 < 1 \\
&\iff |1 + \Delta t \alpha_k|^2 < 1 \quad \text{by (2.6)} \\
&\iff |1 + \Delta t a_k + \Delta t b_k i|^2 < 1 \\
&\iff (1 + \Delta t a_k)^2 + (\Delta t b_k)^2 < 1 \\
&\iff 1 + 2\Delta t a_k + (\Delta t a_k)^2 + (\Delta t b_k)^2 < 1 \\
&\iff 2\Delta t a_k + \Delta t^2 |\alpha_k|^2 < 0 \\
&\iff a_k = \Re e(\alpha_k) < -\frac{\Delta t}{2} |\alpha_k|^2.
\end{aligned}$$

In particular, if iteration (2.5) is convergent, this implies that $\Re e(\alpha_k) < 0$ for all $k = 1, \ldots, N$. By Theorem 2.2 we conclude that the equilibrium solution $\mathbf{u}^*$ of (2.1) is stable.

(b) The first statement follows immediately from Theorem 2.2 and equation (2.6), since the eigenvalue α of $\mathbf{f_u}(\mathbf{u}^*)$ corresponding to μ satisfies $\Re e(\alpha) = \dfrac{\Re e(\mu) - 1}{\Delta t} > 0$ when $\Re e(\mu) > 1$. Next, Theorem 2.2 implies that $\mathbf{u}^*$ loses stability if and only if there is an eigenvalue α of $\mathbf{f_u}$ that crosses the imaginary axis. Since the corresponding eigenvalue of $\mathbf{F_u}$ is $\mu = 1 + \Delta t \alpha$, the second statement follows. ■

Condition (2.7) is due to Shroff and Keller [121]. Note that if iteration (2.5) is convergent, then the equilibrium solution of (2.1) is stable. However, the converse is

not true. Indeed, suppose that all eigenvalues of $\mathbf{f_u}$ have negative real parts, but that for some k we have

$$\Re e(\alpha_k) \geq -\frac{\Delta t}{2}|\alpha_k|^2.$$

This can happen especially if Δt is too large. In this case, the equilibrium solution of (2.1) is stable but iteration (2.5) is not convergent. For a given set of eigenvalues $\{\alpha_k\}$ of $\mathbf{f_u}$, however, the value of Δt can be chosen small enough so that condition (2.7) holds for all k, making the iteration convergent.

This result allows us to calculate bifurcation points of equilibrium solutions of (2.1) by monitoring the eigenvalues of the Jacobian of the time-stepper map. A bifurcation of dynamical system (2.1) will occur when at least one eigenvalue of $\mathbf{F_u}$ crosses the line $\Re e(z) = 1$. If a real eigenvalue μ of $\mathbf{F_u}$ crosses this line through 1, then the corresponding eigenvalue α of $\mathbf{f_u}$ crosses the imaginary axis through 0. In general, this indicates the presence of a turning point. If a complex conjugate pair of eigenvalues of $\mathbf{F_u}$ cross the line $\Re e(z) = 1$, then this corresponds to a Hopf bifurcation point on the solution branch of equilibrium solutions of (2.1). The theory of Hopf bifurcations allows us to estimate the initial period T of solutions on the emanating time-periodic branch. The analog of equation (2.2) for the time-stepper is

$$T \approx \frac{2\pi\Delta t}{\Im m(\mu_{1,2})}, \tag{2.8}$$

where $\mu_{1,2}$ are the dominant pair of eigenvalues crossing the line $\Re e(z) = 1$.

2.1.3 Limitations of Time-Stepper Codes

In Chapter 1 we mentioned that time-steppers suffer from several limitations that make bifurcation-theoretic results difficult, if not impossible, to obtain. In particular:

1. Time-stepper codes cannot compute unstable solutions. The iteration defining the time-stepper can only converge to stable solutions.

2. The rate of convergence of the time-stepper iteration arbitrarily slows down near bifurcation points. As a result, time-stepper codes cannot accurately locate bifurcation points. Instead, an extrapolation process must be used to estimate the location of the bifurcation point. Qualitative changes in the solution may only become apparent farther away from the bifurcation point.

We now explain, using the results from the last section, why these limitations hold for the particular time-stepper used in the Taylor-Couette problem we are studying. The first limitation follows from part (a) of Proposition 2.4: if an equilibrium solution $\mathbf{u}^*$ of (2.1) is unstable, then the time-stepper iteration (2.5) cannot converge to $\mathbf{u}^*$. The fact that the rate of convergence slows down arbitrarily near bifurcation points follows from the estimate (2.4) of Theorem 2.3. To see this, we must consider two cases, depending on how the eigenvalue(s) of the Jacobian matrix $\mathbf{f_u}$ cross the imaginary axis.

First, suppose that a branch of equilibrium solutions of (2.1) undergoes a bifurcation via a single real eigenvalue α of $\mathbf{f_u}$ crossing the imaginary axis through 0, and suppose that the parameter value at which this occurs is $\lambda = \lambda^*$. This means that as $\lambda \to \lambda^*$, we have $\alpha \to 0$, $\alpha \in \mathbb{R}$. Let $\mu = 1 + \Delta t\alpha$ be the corresponding eigenvalue of $\mathbf{F_u}$. Then, as $\lambda \to \lambda^*$, we have $\mu \to 1$. This implies that the spectral radius $\rho(\mathbf{F_u}) \to 1$ as well, and hence the term

$$\rho(\mathbf{F_u}) + \epsilon$$

from estimate (2.4) becomes arbitrarily close to 1. This shows that the rate of convergence of the iteration (2.5) becomes arbitrarily slow near a bifurcation point.

Next, suppose that a branch of equilibrium solutions of (2.1) undergoes a Hopf bifurcation, and that a complex conjugate pair of eigenvalues $\alpha_{1,2}$ of $\mathbf{f_u}$ crosses the imaginary axis through $\pm i\omega, \omega \neq 0$. Let $\mu_{1,2}$ denote the corresponding eigenvalues of $\mathbf{F_u}$. As $\Re e(\alpha_{1,2}) \to 0$, by continuity of the modulus function we conclude that

$|\mu_{1,2}| \to |1 \pm i\Delta t\omega| > 1$. This means that $|\mu_{1,2}|$ becomes arbitrarily close to 1 (and then surpasses 1) as $\alpha_{1,2} \to \pm i\omega$. As in the real eigenvalue case, this implies that the spectral radius approaches 1 and hence the convergence rate of iteration (2.5) becomes arbitrarily slow.

Note that in the case when a real eigenvalue α of $\mathbf{f_u}$ crosses through 0, the time-stepper iteration loses its convergence property at the same time that the equilibrium solution of the dynamical system loses its stability. As α crosses through 0, indicating a loss of stability, the corresponding eigenvalue μ of $\mathbf{F_u}$ crosses through 1, indicating a loss of convergence. The converse is also true: if a real eigenvalue μ of $\mathbf{F_u}$ crosses through 1, then the corresponding eigenvalue α of $\mathbf{f_u}$ crosses through 0.

The situation is slightly more complicated in case of a Hopf bifurcation. When a complex conjugate pair of eigenvalues of $\mathbf{F_u}$ crosses the line $\Re e(z) = 1$ (corresponding to a Hopf bifurcation), their moduli are greater than 1, so the time-stepper iteration is not convergent. However, the converse is not true. Indeed, suppose that iteration (2.5) is not convergent due to a complex pair of eigenvalues $\mu_{1,2}$ of $\mathbf{F_u}$ lying on the unit circle, so that $|\mu_{1,2}| = 1$ and $\Im m(\mu_{1,2}) \neq 0$. Let $\alpha_{1,2}$ be the corresponding eigenvalues of $\mathbf{f_u}$. Repeating the calculation in the proof of part (a) of Proposition 2.4 with equality rather than an inequality, we obtain

$$\Re e(\alpha_{1,2}) = -\frac{\Delta t}{2}|\alpha_{1,2}|^2 < 0.$$

Thus, even though the iteration is no longer convergent, the corresponding equilibrium solution of the dynamical system (2.1) is still stable. Figure 2.1 shows the regions of the complex plane which indicate convergence of the time-stepper (2.5) and stability of equilibrium solutions of (2.1), in terms of the locations of the eigenvalues of the respective Jacobian matrices.

This last situation may seem to be a large drawback in that the iterative scheme will fail to converge before a Hopf bifurcation actually occurs. In practice, however, the situation is not as dire as it first seems. This is because in most applications, the

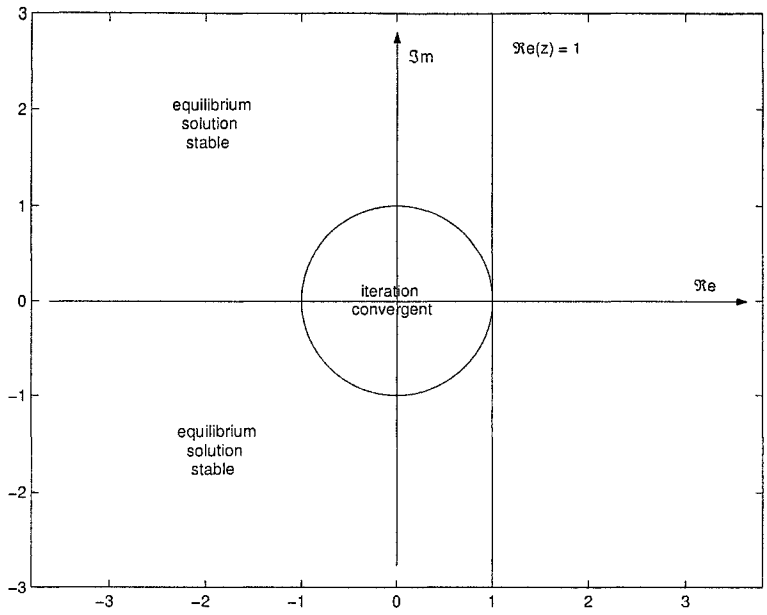

Figure 2.1: *Regions of time-stepper convergence and stability of equilibrium solutions. The time-stepper iteration* $\mathbf{F}$ *is convergent if the spectrum of* $\mathbf{F_u}$ *is contained within the unit circle. An equilibrium solution is stable if the spectrum of* $\mathbf{f_u}$ *lies to the left of the imaginary axis. A bifurcation occurs when an eigenvalue* μ *of* $\mathbf{F_u}$ *crosses the line* $\Re e(z) = 1$.

value of Δt is chosen to be quite small. For example, in the numerical experiments described in Chapter 6, we used $\Delta t = 0.00005$ (in non-dimensional time). This has the effect of rendering the term

$$-\frac{\Delta t}{2}|\alpha_{1,2}|^2$$

quite small, so that in practice, crossing the unit circle (giving loss of convergence of the iteration) and crossing the line $\Re e(z) = 1$ (giving a loss of stability of the equilibrium solution) are nearly simultaneous events.

The Δt term in equation (2.6) unfortunately has an undesirable side effect. The complex mapping $z \mapsto \Delta t z$ is a contraction if $\Delta t < 1$, and in practical applications with very small Δt, this mapping is a very strong contraction. Unless an eigenvalue of $\mathbf{f_u}$ has very large modulus, multiplying it by Δt will give it a very small modulus, and hence the values $\{\Delta t \alpha_k\}$ will be highly clustered about 0. The addition of 1 in

equation (2.6) implies that the spectrum of $\mathbf{F_u}$ will be highly clustered near 1. We will see concrete examples of this clustering exhibited in the numerical results presented in Chapter 6.

This eigenvalue clustering is not only a defect of the explicit Euler time-stepping scheme—the implicit Euler scheme suffers from it as well. The implicit Euler scheme approximates (2.1) by

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{f}(\mathbf{u}^{n+1}, \lambda).$$

This can be rewritten as an iteration of the form

$$\mathbf{u}^{n+1} = (I - \Delta t \mathbf{f})^{-1}\mathbf{u}^n, \quad n = 0, 1, \ldots,$$

an iteration which computes fixed points of $\mathbf{F}(\mathbf{u}) = (I - \Delta t\, \mathbf{f})^{-1}\mathbf{u}$. Differentiating this equation with respect to $\mathbf{u}$, we obtain the Jacobian matrix

$$\mathbf{F_u} = (I - \Delta t\, \mathbf{f_u})^{-1}.$$

Hence, eigenvalues μ of $\mathbf{F_u}$ are related to eigenvalues α of $\mathbf{f_u}$ by

$$\mu = \frac{1}{1 - \Delta t \alpha}.$$

For very small values of Δt, the eigenvalues of $\mathbf{F_u}$ are still clustered near 1.

2.2 Time-Periodic Solutions

In this section, we consider the stability and bifurcations of time-periodic solutions of the dynamical system

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda), \tag{2.9}$$

where as before, $\mathbf{u}(t) \in \mathbb{R}^N$ for each t, $\mathbf{f} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$, and $\mathbf{f}$ is autonomous. First, we define some terms.

A time-periodic solution of (2.9) is a solution $\mathbf{u}(t)$ for which there exists a number $\tau > 0$ such that $\mathbf{u}(t + \tau) = \mathbf{u}(t)$ for all $t \geq 0$. The minimal such τ is called the period

of the solution and is usually denoted by T. In discussing time-periodic solutions, it is helpful to introduce the flow (or trajectory) $\phi(\mathbf{u}_0, t)$, which is defined as the solution $\mathbf{u}(t)$ of (2.9) at time t with given initial condition $\mathbf{u}(0) = \mathbf{u}_0$.

Let $\mathbf{u}^*(t)$ be a time-periodic solution of (2.9) with period T and initial condition $\mathbf{u}^*(0) = \mathbf{u}_0$. The stability of $\mathbf{u}^*(t)$ is determined by the eigenvalues of the Jacobian matrix $\phi_{\mathbf{u}}(\mathbf{u}^*, T)$. This matrix is called the Monodromy matrix and its eigenvalues are termed Floquet multipliers. The main result which governs the stability of $\mathbf{u}^*(t)$ is given below.

Theorem 2.5 (from [120]). *For fixed parameter λ, let $\mathbf{u}^*(t)$ be a time-periodic solution of (2.9) with period T, and let M denote the Monodromy matrix $\phi_{\mathbf{u}}(\mathbf{u}^*, T)$. Denote the N Floquet multipliers by $\mu_1, \ldots, \mu_N$, repeated according to multiplicity. One of them is identically equal to 1, say $\mu_N = 1$. The other $N-1$ Floquet multipliers determine the local stability of $\mathbf{u}^*(t)$ as follows. If $|\mu_k| < 1$ for all $k = 1, \ldots, N-1$, then $\mathbf{u}^*(t)$ is stable. If at least one Floquet multiplier μ_k for some k has modulus greater than 1, then $\mathbf{u}^*(t)$ is unstable.*

Recall that the particular time-stepper for the Taylor-Couette problem we are studying is of the form

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{f}(\mathbf{u}^n, \lambda) \equiv \mathbf{F}(\mathbf{u}^n, \lambda). \tag{2.10}$$

Finding a time-periodic solution of (2.9) amounts to finding a fixed point of the map $\mathbf{F}^k$ for some positive integer k, where $\mathbf{F}^k$ denotes the k-fold composition of $\mathbf{F}$ with itself. The value of k is related to the period T of the time-periodic solution by $T = k\Delta t$. In many numerical experiments, including ours presented in Chapter 6, the value of Δt is very small, which makes the corresponding value of k quite large, typically on the order of several thousand.

The flow function ϕ is a function of both time t and of the initial condition $\mathbf{u}_0$. Another way to interpret this function is as the mapping

$$(\mathbf{u}_0, t) \overset{\phi}{\mapsto} \mathbf{u}(t),$$

where $\mathbf{u}(t)$ satisfies equation (2.9). In terms of the time-stepper $\mathbf{F}$, one would use $\mathbf{u}_0$ as a starting vector and apply iteration (2.10) a total of $k = t/\Delta t$ times to obtain the solution $\mathbf{u}(t)$. Let $\mathbf{G} = \mathbf{F}^k$. Then, it follows that

$$\mathbf{G}(\mathbf{u}_0) = \mathbf{u}(t).$$

That is, the mapping $\mathbf{G}$ and the flow function ϕ are in fact the same. Suppose $\mathbf{u}^*$ is one point on an orbit of a time-periodic solution with period T. Let $k = T/\Delta t$ and let $\mathbf{G} = \mathbf{F}^k$, so that $\mathbf{u}^*$ is a fixed point of $\mathbf{G}$. Then, it follows that eigenvalues of the Jacobian matrix $\mathbf{G}_\mathbf{u}(\mathbf{u}^*)$ are precisely the Floquet multipliers of the Monodromy matrix $\phi_\mathbf{u}(\mathbf{u}^*, T)$.

This allows us to compute bifurcation points on branches of time-periodic solutions simply by monitoring the eigenvalues of the Jacobian matrix of the particular iteration map $\mathbf{G}$. Since the Jacobian matrix $\mathbf{G}_\mathbf{u}$ is in fact the Monodromy matrix, we will refer to its eigenvalues by using the proper term Floquet multipliers. If at least one Floquet multiplier crosses the unit circle in the complex plane, then a bifurcation occurs. The type of bifurcation depends on the number of Floquet multipliers crossing the unit circle and how they cross it. We consider three ways in which this can happen: a real Floquet multiplier can cross at $+1$, a real Floquet multiplier can cross at -1, or a complex conjugate pair of Floquet multipliers can cross at $e^{\pm i\theta}$ for some value of θ that is not an integer multiple of π [61]. Each of these possibilities gives different bifurcation behavior, which we now discuss more fully. Note that we will not address the higher level singularities that result when, for example, two pairs of complex conjugate eigenvalues leave the unit circle simultaneously. Such singularities were not encountered in the experiments we conducted, since we would need to vary two parameters to find these generically.

First, consider the case when a real Floquet multiplier crosses the unit circle at $+1$. Like the equilibrium solution case, this generically indicates that a turning point on the time-periodic solution branch has been reached. However, if the solution possesses some sort of invariance property, a real Floquet multiplier of $+1$ can

also indicate the presence of a symmetry-breaking bifurcation point, just as in the equilibrium case.

Next, suppose that a Floquet multiplier leaves the unit circle by passing through -1. In this case, the time-periodic branch loses stability to a branch of time-periodic solutions with twice the period of the previous branch. This singularity is called a period-doubling bifurcation point.

A complex conjugate pair of Floquet multipliers crossing the unit circle corresponds to a torus bifurcation point [63]. The time-periodic solution branch loses stability to a branch of time-dependent solutions with two incommensurate frequencies. As viewed in phase space, these solutions are no longer simply closed orbits, but rather they are solutions confined to a torus. Note that except in very special cases, these solutions are no longer time-periodic.

2.3 Conclusions

In this chapter, we have explored the inherent limitations that make time-stepper codes ill-suited for bifurcation analysis. In particular, we have seen how the eigenvalues of the linearization of the time-stepper map play a crucial role in determining the location of bifurcation points. Unfortunately, such information is typically not available from a time-stepper code.

The discussion in this chapter illuminates the desirable properties that must be possessed by the numerical methods used to transform the time-stepper code into a bifurcation tool and overcome its inherent limitations:

- The method must be able to compute the dominant eigenvalues of the Jacobian matrix $\mathbf{F_u}$ or of the Monodromy matrix $\boldsymbol{\phi}_\mathbf{u}$. This information can be used to determine the location of bifurcation points more accurately as well as shed light on the nature of the bifurcation (e.g., Hopf bifurcation).

- The method should be able to compute unstable solutions. As we have seen, this means that simple iteration will not suffice. The method must have some means of dealing with the part of the unstable solution corresponding to the eigenvalues or Floquet multipliers with modulus greater than one.

- The rate of convergence of the method should not slow down arbitrarily near bifurcation points. Again, this means that simple iteration will not suffice.

The Newton-Picard methods described in the next chapter possess all of these properties. In addition, these methods enjoy other desirable properties, such as not requiring an explicit construction of the Jacobian or Monodromy matrix.

Chapter 3

Newton-Picard Methods

3.1 Overview

In this chapter, we describe in detail some of the numerical methods that can be used to adapt an existing time-stepper code for numerical bifurcation studies. These methods overcome the inherent limitations possessed by time-stepper codes when they are applied to numerical bifurcation analysis. The methods combine two techniques, a Newton's method and a Picard iterative method, and so they are termed "Newton-Picard methods." As we see later in the chapter, these methods also can serve other purposes, such as accelerating a slowly convergent iteration scheme or computing periodic solutions to systems of ODEs.

In very general terms, the basic idea of these methods is to take the mathematical problem of interest and formulate it as a root-finding problem to which Newton's method can be applied. Rather than using Newton's method on the entire problem, however, the calculation is split into two parts. The first part corresponds to the "slow" or "dangerous" modes of the problem, whose solution is computed in a small-dimensional subspace by direct matrix methods. The other part of the solution is found iteratively by projecting the problem onto the complementary subspace.

This splitting idea was first proposed by Jarausch and Machens [66, 67, 68]. Their method, which they termed "adaptive condensation," is applied to nonlinear systems

of the form

$$A\mathbf{u} = \mathbf{F}(\mathbf{u}), \quad \mathbf{u} \in \mathbb{R}^n,$$

where both the matrix A and the Jacobian matrix $\mathbf{F_u}$ are symmetric. Unfortunately, the hypothesis that both A and $\mathbf{F_u}$ be symmetric restricts the applicability of the adaptive condensation method, a defect remedied by the recursive projection method of Shroff and Keller [121], which is described in detail in the next section. Jarausch [65] later extended the idea to the case of nonsymmetric matrices in systems of parabolic partial differential equations by using a singular subspace (spanned by singular vectors) rather than an invariant subspace (spanned by eigenvectors).

This splitting technique has also been applied to iterative methods for solving linear systems of equations by Burrage, Erhel, and others, who used it to accelerate the convergence of iterative solvers for linear systems [21] and to precondition the restarted GMRES algorithm [51, 20]. This technique has also been used for accelerating convergence of conjugate gradient methods by van Noorden, Lunel, and Bliek [142] and for splitting fast time scales from slow time scales in stiff systems of PDEs by Valorani and Goussis [140].

The remainder of this chapter is devoted to giving more details about the splitting technique alluded to above, and in the process we see how this splitting idea overcomes the time-stepper's inherent limitations. The first splitting method we discuss is the recursive projection method of Shroff and Keller, which is followed by a description of the Newton-Picard method of K. Lust.

3.2 The Recursive Projection Method

The recursive projection method (RPM) was first developed by Shroff and Keller [121], and later expanded somewhat by Keller [74]. Consider a parameter dependent (Picard) iteration scheme of the form

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n, \lambda), \tag{3.1}$$

where $\mathbf{F} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ and N is very large. In the applications of interest to us, this map $\mathbf{F}$ is produced from the discretization of a system of time-dependent PDEs (specifically, the Navier-Stokes equations), and symbolically represents our black-box time-stepper code. The variable n counts the time steps.

Since the map $\mathbf{F}$ is parameter-dependent, it is often of interest to determine how the fixed point $\mathbf{u}^*(\lambda)$ changes as the parameter λ is varied. As explained in the previous chapter, the convergence of the iteration (3.1) slows arbitrarily as an eigenvalue of the Jacobian matrix approaches the unit circle in the complex plane. The iteration diverges if an eigenvalue lies outside the unit circle. The goal of the RPM is to accelerate and stabilize the iteration (3.1) when either of these situations occurs. The RPM accomplishes this by splitting $\mathbb{R}^N$ into an orthogonal direct sum of two subspaces. The first subspace, of small dimension, is the invariant subspace of $\mathbb{R}^N$ that corresponds to these slowly decaying or non-decaying modes. This subspace is spanned by the eigenvectors and generalized eigenvectors of the Jacobian matrix $\mathbf{F}_\mathbf{u}(\mathbf{u}^*(\lambda), \lambda)$ corresponding to the largest eigenvalues. In this subspace, the fixed-point iteration (3.1) is slowly convergent or divergent. Newton's method is therefore used to solve the system obtained by projecting (3.1) onto this small subspace, since the convergence of Newton's method is not sensitive to changes in the spectral radius of the Jacobian. The other subspace is the orthogonal complement of this small-dimensional invariant subspace. In this high-dimension subspace, the projection of the fixed-point iteration still converges and so this iteration is used there. To summarize the RPM in one sentence, the RPM stabilizes a Picard iteration by splitting it into two pieces and replacing the unstable portion with Newton's method.

To make this discussion more mathematically precise, denote the eigenvalues of the Jacobian $\mathbf{F}_\mathbf{u}(\mathbf{u}^*(\lambda), \lambda))$ by $\mu_1(\lambda)$, $\mu_2(\lambda)$, $\ldots, \mu_N(\lambda)$, ordered by decreasing modulus and repeated according to multiplicity. Let $\delta > 0$ be a small, user-defined number, and suppose that $m(\lambda)$ of the eigenvalues have modulus strictly greater than $1 - \delta$ and that no eigenvalue has modulus exactly equal to $1 - \delta$. These $m(\lambda)$

eigenvalues are the ones causing the convergence of the Picard iteration (3.1) to slow down or to diverge. We use the circle of radius $1-\delta$, rather than the unit circle, because we wish to identify those modes that may be causing a slowing of the convergence rate of the iteration. Let $\mathcal{P}$ be the $m(\lambda)$-dimensional subspace of $\mathbb{R}^N$ which is spanned by the eigenvectors and generalized eigenvectors of $\mathbf{F_u}(\mathbf{u}^*(\lambda),\lambda)$ corresponding to $\mu_1,\ldots,\mu_{m(\lambda)}$. Note that $\mathcal{P}$ is an invariant subspace of the Jacobian, that is, $\mathbf{F_u}(\mathbf{u}^*(\lambda),\lambda)\mathcal{P}\subseteq\mathcal{P}$. Let $\mathcal{Q}=\mathcal{P}^\perp$, the orthogonal complement of $\mathcal{P}$ in $\mathbb{R}^N$, and let P and $Q=I-P$ denote the orthogonal projectors onto the subspaces $\mathcal{P}$ and $\mathcal{Q}$, respectively. This gives a direct sum decomposition of $\mathbb{R}^N$ as $\mathbb{R}^N=\mathcal{P}\oplus\mathcal{Q}$, and hence each $\mathbf{u}\in\mathbb{R}^N$ can be written uniquely as

$$\mathbf{u}=\mathbf{p}+\mathbf{q},$$

where $\mathbf{p}=P\mathbf{u}\in\mathcal{P}$ and $\mathbf{q}=Q\mathbf{u}=(I-P)\mathbf{u}\in\mathcal{Q}$. For notational convenience, hereafter we will not explicitly indicate the dependence of the vectors $\mathbf{u},\mathbf{p}$, and $\mathbf{q}$ on the parameter λ, although the reader always should keep this in mind.

If it converges, iteration (3.1) will compute a fixed point of $\mathbf{F}$, i.e., it will find a solution to the equation

$$\mathbf{u}=\mathbf{F}(\mathbf{u},\lambda).$$

Separately applying the projectors P and Q to this equation, we obtain

$$\mathbf{p}=P\mathbf{F}(\mathbf{p}+\mathbf{q},\lambda)\equiv\mathbf{f}(\mathbf{p},\mathbf{q},\lambda) \tag{3.2}$$

and

$$\mathbf{q}=Q\mathbf{F}(\mathbf{p}+\mathbf{q},\lambda)\equiv\mathbf{g}(\mathbf{p},\mathbf{q},\lambda). \tag{3.3}$$

Write $\mathbf{u}^n=\mathbf{p}^n+\mathbf{q}^n$, where $\mathbf{p}^n=P\mathbf{u}^n\in\mathcal{P}$ and $\mathbf{q}^n=Q\mathbf{u}^n\in\mathcal{Q}$. The question that remains is how to obtain $\mathbf{p}^{n+1}$ and $\mathbf{q}^{n+1}$ from $\mathbf{p}^n$ and $\mathbf{q}^n$.

To obtain $\mathbf{p}^{n+1}$, we apply one step of Newton's method to equation (3.2). This requires the solution of

$$(I - \mathbf{f}_{\mathbf{p}}(\mathbf{p}^n, \mathbf{q}^n, \lambda))(\Delta \mathbf{p}^n) = -[\mathbf{p}^n - \mathbf{f}(\mathbf{p}^n, \mathbf{q}^n, \lambda)] \tag{3.4}$$

for $\Delta \mathbf{p}^n$. We then set

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \Delta \mathbf{p}^n. \tag{3.5}$$

Note that all of these computations are done entirely in the small-dimensional subspace $\mathcal{P}$. The Jacobian matrix $\mathbf{f}_{\mathbf{p}}(\mathbf{p}^n, \mathbf{q}^n, \lambda)$ is simply the restriction of the Jacobian $\mathbf{F}_{\mathbf{u}}(\mathbf{u}^n, \lambda)$ to $\mathcal{P}$. Since $\mathcal{P}$ has small dimension, solving for $\Delta \mathbf{p}^n$ can be done using direct linear solvers.

The following lemma will justify our technique for the computation of $\mathbf{q}^{n+1}$. The notation used in the statement and proof of the lemma is cumulative.

Lemma 3.1 (from [121]). *Let $\mathbf{u}^* = \mathbf{p}^* + \mathbf{q}^*$ be a fixed point of $\mathbf{F}$ and let $\mathbf{F}_{\mathbf{u}}^*$ denote the Jacobian matrix $\mathbf{F}_{\mathbf{u}}(\mathbf{u}^*, \lambda)$. Then, the spectral radius of the Jacobian matrix*

$$\mathbf{g}_{\mathbf{q}}^* = \mathbf{g}_{\mathbf{q}}(\mathbf{p}^*, \mathbf{q}^*, \lambda) = Q\mathbf{F}_{\mathbf{u}}^* Q$$

is less than $1 - \delta$.

PROOF: By the Jordan canonical theorem, we know there exists an $N \times N$ invertible matrix W such that

$$\mathbf{F}_{\mathbf{u}}^* = WJW^{-1},$$

where J is the Jordan matrix of $\mathbf{F}_{\mathbf{u}}^*$. Since $\mathcal{P}$ has dimension $m = m(\lambda)$, we can partition this matrix decomposition as

$$\mathbf{F}_{\mathbf{u}}^* = [W_1 \;\; W_2] \begin{bmatrix} J_1 & 0 \\ 0 & J_2 \end{bmatrix} [W_1 \;\; W_2]^{-1} \tag{3.6}$$

where W_1 is $N \times m$, W_2 is $N \times (N - m)$, J_1 is $m \times m$, and J_2 is $(N - m) \times (N - m)$. The matrix J_1 contains all Jordan blocks corresponding to the dominant eigenvalues $\mu_1, \ldots, \mu_m$ while J_2 contains all Jordan blocks corresponding to the remaining $N - m$

eigenvalues, each of which has modulus less than $1-\delta$. Furthermore, the columns of W_1 form a basis for the subspace $\mathcal{P}$, and hence $\mathcal{P} = \mathcal{R}(W_1)$, the range of W_1. In particular, $QW_1 = 0$.

Let $V = [W_1 \;\; QW_2]$, an $N \times N$ matrix. We claim that V is invertible. Since W is nonsingular, we conclude that W_1 has maximal rank m. It remains to be shown that QW_2 has rank $N-m$. Indeed, suppose there is a nonzero vector $\mathbf{w} \in \mathbb{R}^{N-m}$ such that $QW_2\mathbf{w} = \mathbf{0}$. Again since W is nonsingular, W_2 has maximal rank and hence $W_2\mathbf{w} \neq \mathbf{0}$. Because $\mathcal{Q}^{\perp} = \mathcal{P}$, it follows then that $W_2\mathbf{w}$ must be an element of $\mathcal{P} = \mathcal{R}(W_1)$. This contradicts the fact that $W = [W_1 \;\; W_2]$ is nonsingular. Hence, the rank of QW_2 must be $N-m$, from which it follows that V is nonsingular, establishing the claim.

Next, note that since $\mathcal{P} \perp \mathcal{Q}$, we have that $QP = 0$. Since $\mathcal{P}$ is an invariant subspace of $\mathbf{F}_{\mathbf{u}}^*$, it follows that $Q\mathbf{F}_{\mathbf{u}}^*P = 0$ as well. From (3.6), we see that $\mathbf{F}_{\mathbf{u}}^*W_2 = W_2J_2$. Recalling that $P+Q = I$, we obtain

$$QW_2J_2 = Q\mathbf{F}_{\mathbf{u}}^*W_2 = Q\mathbf{F}_{\mathbf{u}}^*(PW_2 + QW_2) = Q\mathbf{F}_{\mathbf{u}}^*QW_2.$$

Since Q is a projection, we know that $Q^2 = Q$. Hence,

$$\begin{aligned} Q\mathbf{F}_{\mathbf{u}}^*QV &= Q\mathbf{F}_{\mathbf{u}}^*Q[W_1 \;\; QW_2] \\ &= [Q\mathbf{F}_{\mathbf{u}}^*QW_1 \;\; Q\mathbf{F}_{\mathbf{u}}^*Q^2W_2] \\ &= [0 \;\; Q\mathbf{F}_{\mathbf{u}}^*QW_2] \\ &= [0 \;\; QW_2J_2] \\ &= [W_1 \;\; QW_2]\begin{bmatrix} 0 & 0 \\ 0 & J_2 \end{bmatrix} \\ &= V\begin{bmatrix} 0 & 0 \\ 0 & J_2 \end{bmatrix}. \end{aligned} \tag{3.7}$$

The invertibility of V allows us to rewrite equation (3.7) as

$$Q\mathbf{F}_{\mathbf{u}}^*Q = V\begin{bmatrix} 0 & 0 \\ 0 & J_2 \end{bmatrix}V^{-1},$$

which is the Jordan canonical form for $Q\mathbf{F}_{\mathbf{u}}^{*}Q$. The eigenvalues of J_2 are $\mu_{m+1}, \ldots, \mu_N$, all of which have moduli less than $1-\delta$. Thus, we obtain the bound on the spectral radius $\rho(Q\mathbf{F}_{\mathbf{u}}^{*}Q) = \rho(\mathbf{g}_{\mathbf{q}}^{*}) < 1-\delta$. ■

The upshot of Lemma 3.1 is that the spectral radius of the Jacobian matrix $\mathbf{g}_{\mathbf{q}}(\mathbf{p}^*, \mathbf{q}^*, \lambda)$ is less than 1. Therefore, we can compute $\mathbf{q}^{n+1}$ by applying one Picard iteration to equation (3.3), knowing that this iteration is convergent in $\mathcal{Q}$. In particular, we have that

$$\mathbf{q}^{n+1} = \mathbf{g}(\mathbf{p}^n, \mathbf{q}^n, \lambda), \tag{3.8}$$

which we then combine with $\mathbf{p}^{n+1}$ from (3.5) to obtain $\mathbf{u}^{n+1} = \mathbf{p}^{n+1} + \mathbf{q}^{n+1}$.

Recall that the iteration (3.1) symbolically represents our time-stepper code. Since we are attempting to treat this code as a black-box, in general we do not have explicit access to the Jacobian matrix. Shroff and Keller's solution is to approximate the Jacobian via a finite-difference approximation to a directional derivative. If $\mathbf{v}$ is any vector of the appropriate dimension, the matrix-vector product $\mathbf{F}_{\mathbf{u}}(\mathbf{u}, \lambda)\mathbf{v}$ is approximated by

$$\mathbf{F}_{\mathbf{u}}(\mathbf{u}, \lambda)\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{u} + \varepsilon\mathbf{v}, \lambda) - \mathbf{F}(\mathbf{u}, \lambda)}{\varepsilon},$$

where ε is determined by the user. The advantage of this approach is that the above computation requires only two applications of the map $\mathbf{F}$ with slightly different arguments. This means we need only call the time-stepper code twice to approximate the action of the Jacobian matrix. To further simplify computations, local coordinates in the small-dimensional subspace $\mathcal{P}$ are used for computing this approximation and for solving (3.4). This also allows for the efficient computation of the eigenvalues of the Jacobian.

As the parameter λ is varied, the fixed point solution $\mathbf{u}^*$ changes, as do the eigenvalues of the Jacobian matrix. If some eigenvalues move closer to the boundary of the unit circle in the complex plane and move outside the circle of radius $1-\delta$, then the dimension of the invariant subspace $\mathcal{P}$ will need to be increased. The RPM

incorporates techniques for monitoring when this is necessary, and recursively builds a basis for the invariant subspace $\mathcal{P}$. It does this by measuring the rate of convergence of the Picard iteration in $\mathcal{Q}$. If this convergence slows down too much, the RPM increases the basis size for $\mathcal{P}$ and computes a new basis. It is also possible that some eigenvalues, originally outside the circle of radius $1 - \delta$, move inside that circle as λ is varied. In this case, the dimension of $\mathcal{P}$ should be decreased. Again, RPM can deal with this situation as well. See [121] or [74] for complete details on how this is accomplished.

As a by-product of monitoring the eigenvalues of the Jacobian, information about linear stability and the location of bifurcation points can be obtained (see Chapter 2). If all of the eigenvalues with modulus greater than $1 - \delta$ still lie within the unit circle, then the solution is a stable solution. However, if one or more lie to the right of the line $\Re e(z) = 1$, then the solution is unstable. Note that, in this case, because the RPM uses Newton's method in $\mathcal{P}$ rather than iteration, this unstable solution can still be computed by the RPM, something not possible with the original time-stepper code. Furthermore, when changing the parameter λ we can monitor any eigenvalues that cross $\Re e(z) = 1$, giving an accurate bifurcation location. By wrapping the RPM around the original time-stepper code, we can perform bifurcation studies without having to rewrite the time-stepper. This is the essence of time-stepper based bifurcation analysis.

There are variations of the RPM algorithm that can be used. Instead of iterating (3.8) once, for instance, the iteration can be performed until convergence to a suitable tolerance (keeping $\mathbf{p}^n$ fixed), with $\mathbf{q}^{n+1}$ denoting the resulting vector. A preconditioner also can be used in conjunction with the RPM, as done by Davidson [37]. The RPM can be used in a numerical path-following situation and can be combined with the pseudo-arclength continuation of Keller [121, 73]. Bošek and Janovský [17] have shown how to extend the RPM to compute the correct orientation in a path-following context. Finally, the Newton-Picard method of K. Lust et al., described in detail in the next section, can be shown to be a generalization of the RPM algorithm.

3.3 The Newton-Picard Method

In this section we consider the Newton-Picard method of K. Lust et al. Several papers [85, 87, 88, 89, 111] have been written on this topic, with varying degrees of detail. The most complete description of the Newton-Picard method can be found in [85]; the paper [88] is also quite useful as it appears to be the most recent. This is the method that we have adapted for use in the Taylor-Couette problem. For distinction, we will call Lust's method "The" Newton-Picard method, even though it is not unique among these methods.

As originally conceived, the Newton-Picard method is designed to efficiently compute time-periodic solutions of a (large) system of ordinary differential equations. Such solutions are often found via a shooting method which incorporates a Newton's method correction [106]. However, for large systems (which can arise by the discretization of a partial differential equation, as in our case), such a method can be prohibitively expensive. As with the RPM, the idea behind the Newton-Picard method is to split the problem into two parts: a direct solver is used in the small-dimensional subspace governed by the most unstable modes, while an approximate solution is found via an iterative Picard method in the orthogonal complement. These two solutions are then combined to yield an approximate solution to the original system of ODE's. An implementation of this algorithm can be found in the software package PDECONT [86].

Since the Newton-Picard method is the one we adapted to act as our wrap-around code for the Taylor-Couette problem, we will describe the method in detail. First, we will discuss the application of the Newton-Picard method to finding equilibrium solutions of systems of ODEs, as outlined in [85]. We also will show how this method, when applied to equilibrium solutions, is in fact a generalization of the RPM of the last section. Then, we will show how the method extends to computing time-periodic solutions.

3.3.1 Equilibrium Solutions

As in Chapter 1, we begin with a system of ODEs of the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda). \tag{3.9}$$

(For convenience, we again have suppressed the dependence of $\mathbf{u}$ on the parameter λ.) Equilibrium solutions of (3.9) satisfy $\mathbf{f}(\mathbf{u}, \lambda) = \mathbf{0}$. Imposing a suitable temporal discretization transforms (3.9) into the iteration

$$\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n, \lambda), \tag{3.10}$$

where n counts the current time step. As discussed in Chapter 2, Adair's code approximates $d\mathbf{u}/dt$ by $(\mathbf{u}^{n+1} - \mathbf{u}^n)/\Delta t$, and hence the iteration mapping is

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \mathbf{f}(\mathbf{u}^n, \lambda) \equiv \mathbf{F}(\mathbf{u}^n, \lambda).$$

Equilibrium solutions $\mathbf{u}^*$ of (3.9) are fixed points of (3.10) and can be computed by repeated iteration of the map $\mathbf{F}$ until successive iterates differ in norm by an amount less than a user-defined tolerance. This is the same situation that the RPM was designed to handle. However, the approach that the Newton-Picard method takes is somewhat different. Rather than finding fixed points of $\mathbf{F}$ by iteration, it applies Newton's method to find roots of the equation $\mathbf{r}(\mathbf{u}) \equiv \mathbf{F}(\mathbf{u}) - \mathbf{u}$. This entails solving the system

$$(M^\nu - I)\Delta\mathbf{u}^\nu = -\mathbf{r}(\mathbf{u}^\nu), \quad \nu = 0, 1, 2, \ldots, \tag{3.11}$$

for $\Delta\mathbf{u}^\nu$, and following this by the update

$$\mathbf{u}^{\nu+1} = \mathbf{u}^\nu + \Delta\mathbf{u}^\nu.$$

Here, the superscript ν counts the Newton iteration, and for convenience we have denoted the Jacobian matrix $\mathbf{F}_\mathbf{u}(\mathbf{u}^\nu, \lambda)$ by M^ν.

Our beginning assumptions are almost identical to those of RPM: Let $\mathbf{u}^*$ be an isolated solution of $\mathbf{r}(\mathbf{u}) = \mathbf{0}$, and let $\mathcal{B}$ be a small neighborhood of $\mathbf{u}^*$. Let

$M(\mathbf{u}) = \mathbf{F}_{\mathbf{u}}(\mathbf{u}, \lambda)$ for $\mathbf{u} \in \mathcal{B}$ and denote its eigenvalues by $\mu_1, \ldots, \mu_N$, ordered by decreasing modulus and repeated according to multiplicity. (This notation is not complete in that it suppresses the dependence of the eigenvalues on $\mathbf{u}$ and on λ, but for simplicity this notation will suffice.) Assume that for every $\mathbf{u} \in \mathcal{B}$, precisely p eigenvalues of $M(\mathbf{u})$ lie outside the circle of radius $1 - \delta$ in the complex plane, where as before $\delta > 0$ is a small number determined by the user. Assume further that no eigenvalue has modulus equal to $1 - \delta$. Hence, for every $\mathbf{u} \in \mathcal{B}$, we have

$$|\mu_1| \geq |\mu_2| \geq \cdots \geq |\mu_p| > 1 - \delta > |\mu_{p+1}| \geq \cdots \geq |\mu_N|.$$

As Lust points out in [85], page 48, this assumption is typically observed in practice.

As with the RPM, let $\mathcal{P}$ denote the p-dimensional invariant subspace of the Jacobian $M(\mathbf{u})$ corresponding to the eigenvalues $\mu_1, \ldots, \mu_p$. Let V_p be an $N \times p$ real matrix whose columns form an orthonormal basis for $\mathcal{P}$. These basis vectors can be found via a real Schur decomposition of $M(\mathbf{u})$. Next, let $\mathcal{Q} = \mathcal{P}^{\perp}$ and suppose that V_q is a real $N \times (N - p)$ matrix whose columns form an orthonormal basis for $\mathcal{Q}$.

Let $P = V_p V_p^T$ and $Q = V_q V_q^T = I - P$ be orthogonal projectors of $\mathbb{R}^N$ onto $\mathcal{P}$ and $\mathcal{Q}$, respectively. For any $\mathbf{u} \in \mathbb{R}^N$, this gives the unique decomposition

$$\mathbf{u} = P\mathbf{u} + Q\mathbf{u},$$

where $P\mathbf{u} \in \mathcal{P}$ and $Q\mathbf{u} \in \mathcal{Q}$. It will often be convenient to work with local coordinates in $\mathcal{P}$ and $\mathcal{Q}$. Utilizing the matrices V_p and V_q, we easily get a change of coordinates by setting

$$\bar{\mathbf{p}} = V_p^T \mathbf{u} \in \mathbb{R}^p, \qquad \bar{\mathbf{q}} = V_q^T \mathbf{u} \in \mathbb{R}^{N-p},$$

from which we have $\mathbf{u} = V_p \bar{\mathbf{p}} + V_q \bar{\mathbf{q}}$. Similarly, we can write

$$\Delta\mathbf{u} = V_p \Delta\bar{\mathbf{p}} + V_q \Delta\bar{\mathbf{q}}.$$

Substituting this decomposition into the Newton's method equation (3.11) and multiplying by $\begin{bmatrix} V_q & V_p \end{bmatrix}^T$ yields the system

$$\begin{bmatrix} V_q^T(M^\nu - I)V_q & 0 \\ V_p^T M^\nu V_q & V_p^T(M^\nu - I)V_p \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{q}}^\nu \\ \Delta\bar{\mathbf{p}}^\nu \end{bmatrix} = - \begin{bmatrix} V_q^T \mathbf{r}(\mathbf{u}^\nu) \\ V_p^T \mathbf{r}(\mathbf{u}^\nu) \end{bmatrix}. \tag{3.12}$$

Here we have used the fact that $V_p^T V_q = V_q^T V_p = 0$ since $\mathcal{P}$ and $\mathcal{Q}$ are orthogonal to each other, and that $V_q^T M^\nu V_p = 0$ since $\mathcal{P}$ is an invariant subspace of M^ν. Note that system (3.12) is block lower triangular. This enables us to more easily solve system (3.12) for $\Delta\bar{\mathbf{p}}^\nu$ and $\Delta\bar{\mathbf{q}}^\nu$, from which we obtain

$$\bar{\mathbf{p}}^{\nu+1} = \bar{\mathbf{p}}^\nu + \Delta\bar{\mathbf{p}}^\nu$$
$$\bar{\mathbf{q}}^{\nu+1} = \bar{\mathbf{q}}^\nu + \Delta\bar{\mathbf{q}}^\nu.$$

We then calculate $\mathbf{u}^{\nu+1}$ by

$$\mathbf{u}^{\nu+1} = V_p\bar{\mathbf{p}}^{\nu+1} + V_q\bar{\mathbf{q}}^{\nu+1}.$$

The basic idea behind the Newton-Picard method is to avoid solving (3.12) directly, since in many applications (including ours) this is a very large system and hence quite expensive to solve using direct methods. Instead, the Newton-Picard method finds $\Delta\bar{\mathbf{q}}^\nu$ via a Picard iteration. In local coordinates, the equations for computing $\Delta\bar{\mathbf{p}}^\nu$ form a small system which can be solved by direct methods. We now turn to the details of computing $\Delta\bar{\mathbf{q}}^\nu$ and $\Delta\bar{\mathbf{p}}^\nu$.

Solution of $\Delta\bar{\mathbf{q}}^\nu$

From (3.12), the system that determines $\Delta\bar{\mathbf{q}}^\nu$ is

$$(V_q^T M^\nu V_q - I)\Delta\bar{\mathbf{q}}^\nu = -V_q^T \mathbf{r}(\mathbf{u}^\nu). \tag{3.13}$$

Hereafter, we refer to system (3.13) as the $\mathcal{Q}$-equations. This is a large system that we wish to avoid solving directly. Since the spectral radius of $V_q^T M^\nu V_q$ is less than one by Lemma 3.1, we know that the Neumann series for $(V_q^T M^\nu V_q - I)^{-1}$ is convergent and hence

$$(V_q^T M^\nu V_q - I)^{-1} = -\sum_{k=0}^{\infty}(V_q^T M^\nu V_q)^k.$$

If we approximate this series by taking the first ℓ terms, we get the approximation

$$\Delta\bar{\mathbf{q}}^\nu \approx \sum_{k=0}^{\ell-1}(V_q^T M^\nu V_q)^k V_q^T \mathbf{r}(\mathbf{u}^\nu).$$

We can view this equation as the following Picard iteration (dropping the superscript ν for the moment):

$$\begin{aligned} \Delta\bar{\mathbf{q}}^{[0]} &= \mathbf{0}, \\ \Delta\bar{\mathbf{q}}^{[k]} &= V_q^T M V_q \Delta\bar{\mathbf{q}}^{[k-1]} + V_q^T \mathbf{r}, \quad k = 1, \ldots, \ell, \end{aligned} \tag{3.14}$$

and then setting $\Delta\bar{\mathbf{q}}^{\nu} = \Delta\bar{\mathbf{q}}^{[\ell]}$. Note that in (3.14) we have used the superscripts $[k]$ to count the Picard iteration number, *not* the Newton step number of (3.12).

Notice that the large matrix V_q still appears in equation (3.14). Multiplying (3.14) by V_q and recalling that $V_q\bar{\mathbf{q}} = \mathbf{q} \in \mathbb{R}^N$ yields the Picard iteration

$$\begin{aligned} \Delta\mathbf{q}^{[0]} &= \mathbf{0}, \\ \Delta\mathbf{q}^{[k]} &= QM\Delta\mathbf{q}^{[k-1]} + Q\mathbf{r}, \quad k = 1, \ldots, \ell. \\ \Delta\mathbf{q}^{\nu} &= \Delta\mathbf{q}^{[\ell]}. \end{aligned} \tag{3.15}$$

Now (3.15) only involves the projector Q. Since $Q = I - P = I - V_p V_p^T$, we need only use the (small) matrix V_p when executing (3.15). In fact, for numerical stability the matrix V_p is not used directly; rather, the projection onto $\mathcal{Q}$ is performed via a modified Gram-Schmidt method (without normalization).

Using a larger value of ℓ should aid the convergence rate of the Picard iteration (3.15), but there are additional costs involved in doing so (see the next paragraph). Lust et al. [89] state that in initial tests of the Newton-Picard method, $\ell = 1$ was usually sufficient. For robustness, Lust later [88] employs a variable number of Picard iterations. This is done by measuring the norm of the residual

$$||QM\Delta\mathbf{q} + Q\mathbf{r} - \Delta\mathbf{q}||$$

and halting the iteration when this norm falls below a user-defined tolerance. Note that this norm is simply the norm of the difference of successive Picard iterates.

Observe that for $\ell > 1$, equation (3.15) requires that we form the matrix-vector product $M^{\nu}\Delta\mathbf{q}$. Recall that M^{ν} is the Jacobian matrix $\mathbf{F}_{\mathbf{u}}(\mathbf{u}^{\nu}, \lambda)$. To compute a matrix-vector product of the form $M^{\nu}\mathbf{v}$, a finite difference approximation can be

made, as Shroff and Keller suggested with RPM. As with the RPM, using a finite difference approximation of the Jacobian requires two calls to the time-stepper code, which is an additional cost of using the Picard iteration (3.15) with $\ell \geq 2$ that is not present in the $\ell = 1$ case.

As a final note, Lust mentions that other iterative methods can be used to solve the $\mathcal{Q}$-equations (3.13). Specifically mentioned is GMRES, a detailed analysis of which is made in [85]. However, the storage requirements for such a method are much larger than the simple Picard iteration given above. For this reason GMRES is not used in our work.

Solution of $\Delta\bar{\mathbf{p}}^{\nu}$

The system of equations that defines $\Delta\bar{\mathbf{p}}^{\nu}$ can be written as

$$V_p^T(M^{\nu} - I)V_p\Delta\bar{\mathbf{p}}^{\nu} = -V_p^T\mathbf{r}(\mathbf{u}^{\nu}) - V_p^T M^{\nu} V_q \Delta\bar{\mathbf{q}}^{\nu}, \tag{3.16}$$

which we will hereafter refer to as the $\mathcal{P}$-equations. Note that the updated value of $\Delta\bar{\mathbf{q}}^{\nu}$ computed by (3.15) is used to compute $\Delta\bar{\mathbf{p}}^{\nu}$. We can think of this as a Gauss-Seidel-like approach to solving for $\Delta\bar{\mathbf{p}}^{\nu}$.

Since the $\mathcal{P}$-equations (3.16) form a small (p-dimensional) system, we can solve for $\Delta\bar{\mathbf{p}}^{\nu}$ using any method; cost is not an issue. Unfortunately, the system (3.16) is singular at bifurcation and fold points, and is very ill-conditioned near these points. For these reasons, Lust et al. [88] suggest using a least squares method based on the singular value decomposition to solve (3.16). This has the advantage of being numerically stable [138].

Computing a Basis of $\mathcal{P}$

One of the key computations necessary for the Newton-Picard method's success is the computation of an orthonormal basis of $\mathcal{P}$, the vectors of which are stored as the columns of the matrix V_p. Since the invariant subspace $\mathcal{P}$ corresponds to the eigenvalues of the Jacobian matrix M with largest modulus, it is logical to use a

subspace iteration algorithm [127] to compute a basis for $\mathcal{P}$. To ensure robustness of their algorithm, Lust et al., use a very sophisticated version of subspace iteration, namely subspace iteration with projection and locking (deflation). Saad [113] gives more complete details of this algorithm as well as convergence proofs.

Above we described the invariant subspace $\mathcal{P}$ as the subspace spanned by the (generalized) eigenvectors of the Jacobian corresponding to the dominant eigenvalues. While this is mathematically accurate, these eigenvectors are not actually used to form the matrix V_p. Instead, at each step of the subspace iteration algorithm, a real Schur decomposition is computed, and it is these Schur vectors that, upon convergence of the subspace iteration, are used to form an orthonormal basis of $\mathcal{P}$. As Saad [113] points out, Schur vectors can be obtained in a more numerically stable way than can eigenvectors, and they are less sensitive to rounding errors.

3.3.2 Comparison of the RPM and the Newton-Picard Method for Equilibrium Solutions

In the case of finding fixed points of $\mathbf{F}$, the recursive projection method and the Newton-Picard method are doing essentially the same thing: using a direct solver in a small-dimensional subspace and an iterative scheme in its orthogonal complement. However, the approaches taken by the two methods are different. The RPM begins with a Picard iteration and stabilizes it by using Newton's method. The Newton-Picard method begins with Newton's method and makes it more efficient by using a Picard iteration. However, the Newton-Picard method, when applied to a fixed-point iteration, is actually a generalization of the RPM in the following sense. Recall that the Picard iteration portion of the Newton-Picard method is given by

$$\begin{aligned}
\Delta\mathbf{q}^{[0]} &= \mathbf{0}, \\
\Delta\mathbf{q}^{[k]} &= QM\Delta\mathbf{q}^{[k-1]} + Q\mathbf{r}, \quad k = 1, \ldots, \ell, \\
\Delta\mathbf{q}^{\nu} &= \Delta\mathbf{q}^{[\ell]}.
\end{aligned}$$

For $\ell = 1$, this yields

$$\Delta \mathbf{q}^{\nu} = \Delta \mathbf{q}^{[1]} = QM \underbrace{\Delta \mathbf{q}^{[0]}}_{=0} + Q\mathbf{r} = Q(\mathbf{F}(\mathbf{u}^{\nu}) - \mathbf{u}^{\nu}),$$

and hence,

$$\mathbf{q}^{\nu+1} = \mathbf{q}^{\nu} + \Delta \mathbf{q}^{\nu} = Q\mathbf{u}^{\nu} + Q(\mathbf{F}(\mathbf{u}^{\nu}) - \mathbf{u}^{\nu}) = Q\mathbf{F}(\mathbf{u}^{\nu}).$$

This is simply the projection onto $\mathcal{Q}$ of the original Picard iteration $\mathbf{u}^{n+1} = \mathbf{F}(\mathbf{u}^n, \lambda)$. Comparing this with the RPM, we see that the Picard iteration of the Newton-Picard method, using $\ell = 1$, is precisely the fixed-point iteration of the RPM in $\mathcal{Q}$.

A closer analysis of the Newton-Picard $\mathcal{P}$-equations

$$V_p^T(M^{\nu} - I)V_p \Delta \bar{\mathbf{p}}^{\nu} = -V_p^T \mathbf{r}(\mathbf{u}^{\nu}) - V_p^T M^{\nu} V_q \Delta \bar{\mathbf{q}}^{\nu}$$

shows another difference between the RPM and the Newton-Picard method. Recall that the Newton-Picard method can be considered as a Gauss-Seidel-like approach, since the updated value of $\Delta \bar{\mathbf{q}}^{\nu}$ is used in the computation of $\Delta \bar{\mathbf{p}}^{\nu}$. However, if we set the term $V_p^T M^{\nu} V_q$ to zero, then the $\mathcal{P}$-equations are completely decoupled from the $\mathcal{Q}$-equations, yielding a what can be thought of as a Jacobi-like method. With $\ell = 1$ in the Picard iteration for the $\mathcal{Q}$-equations, such a Jacobi-like variant of the Newton-Picard method is actually identical to the RPM. However, Lust [85] states that in practice the term $V_p^T M^{\nu} V_q$ can be quite large. Even though the Gauss-Seidel and Jacobi variants have the same asymptotic convergence rates, because the term $V_p^T M^{\nu} V_q$ is omitted from the Jacobi-like scheme, it needs a more accurate basis for $\mathcal{P}$ than does the Gauss-Seidel-like scheme to exhibit a similar performance (in terms of the number of Newton-Picard iterations required).

To determine whether it is necessary to increase the size of the basis of $\mathcal{P}$, the RPM measures the speed of convergence of the iteration in $\mathcal{Q}$. Unfortunately, this speed also can be affected by an inaccurate basis for $\mathcal{P}$ or a bad initial guess for the solution. In contrast, the Newton-Picard method's criterion for determining the basis

size is much more robust and involves actually estimating the eigenvalues with largest modulus of the Jacobian matrix.

Finally, recall that the RPM was developed to stabilize a fixed-point iteration scheme. If another type of problem is to be solved (not a fixed point problem), then significant modifications may need to be done in order for the RPM to work. It is not clear that the RPM is general enough to be applied in many different situations. However, since the Newton-Picard method was designed to simplify Newton's method, it can be used on the wide variety of problems to which Newton's method can be applied. In particular, the Newton-Picard method can easily be extended to computing time-periodic solutions, which we now describe.

3.3.3 Time-Periodic Solutions

Let us turn to the application for which the method was originally intended, namely the computation of time-periodic solutions to a nonlinear system of ODEs. As before, we consider a parameter-dependent system of the form

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda). \tag{3.17}$$

In the discussion that follows, we assume that the parameter λ is held fixed, and our notation will suppress the dependence of variables on λ. We further assume that $\mathbf{f}$ is autonomous, that is, it does not explicitly depend on time t. This assumption implies that a periodic solution of the system is entirely determined by the initial condition $\mathbf{u}(0) = \mathbf{u}_0$ and the period of the solution, which we denote by T. It is these two quantities which the Newton-Picard method computes.

Note that if $\mathbf{u}(t)$ is a periodic solution of (3.17), then so is $\mathbf{u}(t+\tau)$ for any $\tau \in \mathbb{R}$. Hence, there is a non-uniqueness associated with periodic solutions. Essentially, any point of the orbit (as viewed in phase space) can be viewed as the point at time 0. We are therefore free to choose this point however we like by introducing an auxiliary equation $\psi(\mathbf{u}_0, T) = 0$, which is called a phase condition. Examples of

phase conditions used in practice can be found in [106] and [120]. Hence, a periodic solution can be found by solving the two-point boundary value problem

$$\begin{cases} \frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda), \\ \frac{dT}{dt} = 0, \\ \mathbf{u}(T) = \mathbf{u}_0, \\ \psi(\mathbf{u}_0, T) = 0. \end{cases} \tag{3.18}$$

Let $\boldsymbol{\phi}(\mathbf{u}_0, t)$ denote the solution of the initial value problem (3.17) at time t for a given initial condition $\mathbf{u}_0$ at $t = 0$. This solution $\boldsymbol{\phi}$ is found by the time-stepper code, stepping forward in time until time t is reached. A standard technique to solve the two-point boundary value problem (3.18) is to use a shooting approach. As detailed in [106], a shooting approach involves using Newton's method to solve the system

$$\begin{cases} \mathbf{r}(\mathbf{u}_0, T) \equiv \boldsymbol{\phi}(\mathbf{u}_0, T) - \mathbf{u}_0 = \mathbf{0}, \\ \psi(\mathbf{u}_0, T) = 0. \end{cases} \tag{3.19}$$

In particular, at each Newton iteration we solve the system

$$\begin{bmatrix} M^\nu - I & \boldsymbol{\phi}_T^\nu \\ \psi_{\mathbf{u}}^\nu & \psi_T^\nu \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}_0^\nu \\ \Delta T^\nu \end{bmatrix} = - \begin{bmatrix} \mathbf{r}(\mathbf{u}_0^\nu, T^\nu) \\ \psi(\mathbf{u}_0^\nu, T^\nu) \end{bmatrix}, \tag{3.20}$$

and then set

$$\mathbf{u}_0^{\nu+1} = \mathbf{u}_0^\nu + \Delta \mathbf{u}_0^\nu,$$
$$T^{\nu+1} = T^\nu + \Delta T^\nu.$$

As before, the superscript ν counts the Newton iterations, the matrix M is the Jacobian $d\boldsymbol{\phi}/d\mathbf{u}$, and all derivatives in (3.20) are evaluated at $(\mathbf{u}_0^\nu, T^\nu)$. As in the equilibrium case, we can approximate the action of M^ν on a vector $\mathbf{v}$ by a finite difference:

$$M^\nu \mathbf{v} \approx \frac{\boldsymbol{\phi}(\mathbf{u}_0^\nu + \varepsilon \mathbf{v}, T^\nu) - \boldsymbol{\phi}(\mathbf{u}_0^\nu, T^\nu)}{\varepsilon}.$$

Recall that this evaluation of $M^\nu \mathbf{v}$ will require two calls to the time-stepper code, with slightly different initial conditions. Since $\boldsymbol{\phi}(\mathbf{u}_0, t)$ solves $d\mathbf{u}/dt = \mathbf{f}(\mathbf{u})$ by definition, we easily conclude that

$$\boldsymbol{\phi}_T^\nu = \mathbf{f}(\boldsymbol{\phi}(\mathbf{u}_0^\nu, T^\nu)).$$

The evaluation of the two derivatives $\psi_{\mathbf{u}}^{\nu}$ and ψ_{T}^{ν} depends on the form of the phase condition $\psi(\mathbf{u}_0, T)$ used.

Solving the system (3.20) directly is expensive when N is large, as it will be in our case. Hence, as in the equilibrium case, we apply the Newton-Picard method to solve (3.20). We make an assumption analogous to that of the equilibrium case: let $(\mathbf{u}_0^*, T^*)$ be an isolated solution of (3.19), let $\mathcal{B}$ be a small neighborhood of $(\mathbf{u}_0^*, T^*)$, and let $M(\mathbf{u}_0, T) = \boldsymbol{\phi}_{\mathbf{u}}(\mathbf{u}_0, T)$ for $(\mathbf{u}_0, T) \in \mathcal{B}$. Denote the eigenvalues of M by μ_i, $i = 1, \ldots, N$, and assume that for all $(\mathbf{u}_0, T) \in \mathcal{B}$ precisely p eigenvalues lie outside the circle of radius $1 - \delta$ for some small $\delta > 0$. As in the equilibrium case, the parameter δ is determined by the user.

As before, let $\mathcal{P}$ denote the invariant subspace of M spanned by the eigenvectors and generalized eigenvectors of M which correspond to the eigenvalues which lie outside the circle of radius $1 - \delta$. Let V_p be an orthogonal matrix whose columns span $\mathcal{P}$; the orthogonal projector onto $\mathcal{P}$ is given by $P = V_p V_p^T$. Also, let $\mathcal{Q} = \mathcal{P}^{\perp}$ with orthonormal basis V_q and orthogonal projector $Q = V_q V_q^T = I - P$. For $\Delta\mathbf{u}_0 \in \mathbb{R}^N$, this yields the unique decomposition

$$\Delta\mathbf{u}_0 = V_p \Delta\bar{\mathbf{p}} + V_q \Delta\bar{\mathbf{q}}, \quad \text{where} \quad \Delta\bar{\mathbf{p}} \in \mathbb{R}^p \text{ and } \Delta\bar{\mathbf{q}} \in \mathbb{R}^{N-p}.$$

Substituting this decomposition into the system (3.20) and multiplying the first N equations by $\begin{bmatrix} V_q & V_p \end{bmatrix}^T$, we obtain the system

$$\begin{bmatrix} V_q^T(M^{\nu} - I)V_q & 0 & V_q^T \boldsymbol{\phi}_T^{\nu} \\ V_p^T M^{\nu} V_q & V_p^T(M^{\nu} - I)V_p & V_p^T \boldsymbol{\phi}_T^{\nu} \\ \psi_{\mathbf{u}}^{\nu} V_q & \psi_{\mathbf{u}}^{\nu} V_p & \psi_T^{\nu} \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{q}}^{\nu} \\ \Delta\bar{\mathbf{p}}^{\nu} \\ \Delta T^{\nu} \end{bmatrix} = - \begin{bmatrix} V_q^T \mathbf{r}(\mathbf{u}_0^{\nu}, T^{\nu}) \\ V_p^T \mathbf{r}(\mathbf{u}_0^{\nu}, T^{\nu}) \\ \psi(\mathbf{u}_0^{\nu}, T^{\nu}) \end{bmatrix}. \quad (3.21)$$

As in the equilibrium case, we have used the fact that $V_p^T V_q = V_q^T V_p = V_q^T M^{\nu} V_p = 0$. Once $\Delta\bar{\mathbf{q}}^{\nu}$, $\Delta\bar{\mathbf{p}}^{\nu}$, and ΔT^{ν} have been found, we set

$$\mathbf{u}_0^{\nu+1} = \mathbf{u}_0^{\nu} + V_p \Delta\bar{\mathbf{p}}^{\nu} + V_q \Delta\bar{\mathbf{q}}^{\nu}$$

$$T^{\nu+1} = T^{\nu} + \Delta T^{\nu}.$$

The vector

$$\mathbf{b}^* \equiv \left.\frac{\partial \boldsymbol{\phi}}{\partial T}\right|_{(\mathbf{u}_0^*, T^*)}$$

is an eigenvector of the Monodromy matrix $M(\mathbf{u}_0^*, T^*)$ corresponding to the eigenvalue 1 [85]. Hence, $\mathbf{b}^* \in \mathcal{P}$ by our assumption above, which implies that $V_q^T \mathbf{b}^* = 0$. Thus, the term $V_q^T \boldsymbol{\phi}_T^\nu$ appearing in equation (3.21) is usually quite small and converges to zero as the Newton iteration converges. Lust et al., therefore neglect this term, and so will we.

With this simplification, the solution of (3.21) is computed as in the equilibrium case. We first solve

$$V_q^T (M^\nu - I) V_q \Delta\bar{\mathbf{q}}^\nu = -V_q^T \mathbf{r}(\mathbf{u}_0^\nu, T^\nu)$$

for $\Delta\bar{\mathbf{q}}^\nu$ via a Picard iteration, either using a fixed number ℓ steps or by iterating until convergence to a desired tolerance is reached. Next, $\Delta\bar{\mathbf{p}}^\nu$ and ΔT^ν are found by directly solving the small system

$$\begin{bmatrix} V_p^T (M^\nu - I) V_p & V_p^T \boldsymbol{\phi}_T^\nu \\ \psi_{\mathbf{u}}^\nu V_p & \psi_T^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T [\mathbf{r}(\mathbf{u}_0^\nu, T^\nu) + M^\nu V_q \Delta\bar{\mathbf{q}}^\nu] \\ \psi(\mathbf{u}_0^\nu, T^\nu) + \psi_{\mathbf{u}}^\nu V_q \Delta\bar{\mathbf{q}}^\nu \end{bmatrix}.$$

The basis V_p is computed in precisely the same way as in the equilibrium case.

Note that this use of the Newton-Picard method has no analog in the recursive projection method. The RPM in general cannot easily compute periodic solutions to (3.17), although it has been used to find travelling wave solutions [84].

3.4 Continuation Around Turning Points

In the previous sections, we have suppressed the dependence on the parameter λ for notational convenience. Our primary interest, however, is in observing solution bifurcations as λ is varied, so in this section the dependence on the parameter λ will be made explicit. We consider the system of ordinary differential equations

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda) \tag{3.22}$$

as our basic equation for both the equilibrium and time-periodic cases.

Often, the goal of numerical continuation is to determine the entire solution set

$$G = \left\{ (\mathbf{u}, \lambda) \in \mathbb{R}^{N+1} \;\middle|\; \frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}, \lambda) \right\}.$$

Of course, an entire continuous solution branch cannot be computed; rather, a discrete set of points of G are found. The main idea of continuation is to produce a procedure whereby, given a solution $(\mathbf{u}_0, \lambda_0) \in G$, a nearby point of G can be computed.

A simple approach to this end is first to compute the solution $(\mathbf{u}_0, \lambda_0) \in G$ for a particular parameter value λ_0, change the parameter by some $\Delta\lambda$ to obtain $\lambda_1 = \lambda_0 + \Delta\lambda$, and then compute the solution $(\mathbf{u}_1, \lambda_1) \in G$. This process is repeated to find a set of solutions $(\mathbf{u}_i, \lambda_i) \in G$, $i = 0, 1, 2, \ldots$. This method, called parameter continuation, works well at regular points, that is, at those points where the Jacobian matrix $\mathbf{f}_\mathbf{u}$ is nonsingular. This is essentially the method employed by the TAY code, which has two parameters, R_i and R_o, of which only one is allowed to vary at a time.

At a turning point, however, parameter continuation breaks down, for at such points the solution $\mathbf{u}$ is not a function of λ. Because the Jacobian matrix $\mathbf{f}_\mathbf{u}$ is singular at a turning point, the Implicit Function Theorem fails to guarantee a unique solution branch as a function of λ at the turning point—the solution is not a single-valued function of λ. If the value of λ is increased past the turning point, either the numerical scheme used to solve (3.22) will not converge or will converge to a solution on another branch. Parameter continuation can break down at other points, but we will focus on only turning points in this section.

The remedy is to introduce a new parameter s by augmenting the defining system with a new equation that defines this new parameter. This parameterization equation should be chosen so that the solution is single-valued in s and the Jacobian matrix of this augmented system is nonsingular at turning points. One of the most widely and successfully used parameterizations is the arclength continuation method of H. B. Keller [73]. In this method, a parameter s is introduced that represents arclength (or an approximation of it) along the curve as measured from a given solution $(\mathbf{u}_0, \lambda_0)$.

Both $\mathbf{u}$ and λ are viewed as unknown functions of this parameter s. Parameter continuation, as described above, is carried out with s instead of λ: at each value s_i, we compute the solution $(\mathbf{u}(s_i), \lambda(s_i)) \in G$. This method allows us to follow a solution branch around a turning point.

Both the RPM and the Newton-Picard method can be extended to include the use of such a parameterization equation. Since the Newton-Picard method is a generalization of the RPM, we describe the use of a parameterization equation only for the Newton-Picard method in the next two sections.

3.4.1 Equilibrium Solutions

As in Section 3.3.1, the goal is to find an equilibrium solution of (3.22) by computing a fixed point of the iteration $\mathbf{F}(\mathbf{u}^n(\lambda)) = \mathbf{u}^{n+1}(\lambda)$. Instead of using the iteration, however, we apply Newton's method to solving $\mathbf{r}(\mathbf{u}(\lambda)) = \mathbf{F}(\mathbf{u}(\lambda)) - \mathbf{u}(\lambda)$. Let us introduce a parameterization equation of the form $n(\mathbf{u}, \lambda, s) = 0$, where s is the new parameter. In particular, we have in mind Keller's arclength continuation, but the analysis in this section and the next applies to any parameterization equation. Both $\mathbf{u}$ and λ are treated as functions of s. For fixed s, the system we wish to solve by Newton's method is

$$\begin{bmatrix} \mathbf{F}(\mathbf{u}(s), \lambda(s)) - \mathbf{u}(s) \\ n(\mathbf{u}(s), \lambda(s), s) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}. \tag{3.23}$$

For each Newton iteration, this requires the solution of the system

$$\begin{bmatrix} \mathbf{F}_{\mathbf{u}}(\mathbf{u}^\nu, \lambda^\nu) - I & \mathbf{F}_\lambda(\mathbf{u}^\nu, \lambda^\nu) \\ n_{\mathbf{u}}(\mathbf{u}^\nu, \lambda^\nu, s) & n_\lambda(\mathbf{u}^\nu, \lambda^\nu, s) \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} \mathbf{r}(\mathbf{u}^\nu, \lambda^\nu) \\ n(\mathbf{u}^\nu, \lambda^\nu, s) \end{bmatrix}, \tag{3.24}$$

for $\Delta\mathbf{u}^\nu$ and $\Delta\lambda^\nu$. We then set $\mathbf{u}^{\nu+1} = \mathbf{u}^\nu + \Delta\mathbf{u}^\nu$ and $\lambda^{\nu+1} = \lambda^\nu + \Delta\lambda^\nu$. As before, we will denote $\mathbf{F}_{\mathbf{u}}(\mathbf{u}^\nu, \lambda^\nu)$ by M^ν. For notational convenience we will denote $\mathbf{F}_\lambda(\mathbf{u}^\nu, \lambda^\nu)$ by $\mathbf{F}_\lambda^\nu$, and similarly for the other terms appearing in (3.24).

Utilizing the basis V_p for the subspace $\mathcal{P}$ and the orthogonal projector $P = V_p V_p^T$ onto $\mathcal{P}$, and similarly V_q and Q for the subspace $\mathcal{Q}$, we can write $\Delta\mathbf{u} = \Delta\mathbf{p} + \Delta\mathbf{q}$,

where $\Delta\mathbf{p} \in \mathcal{P}$ and $\Delta\mathbf{q} \in \mathcal{Q}$. Setting $\Delta\bar{\mathbf{p}} = V_p^T \Delta\mathbf{u} \in \mathbb{R}^p$ and $\Delta\bar{\mathbf{q}} = V_q^T \mathbf{u} \in \mathbb{R}^{N-p}$ in local coordinates, we can also write

$$\Delta\mathbf{u} = V_p \Delta\bar{\mathbf{p}} + V_q \Delta\bar{\mathbf{q}}.$$

Substituting this decomposition of $\Delta\mathbf{u}$ into (3.24) and multiplying the first N equations by $\begin{bmatrix} V_q & V_p \end{bmatrix}^T$ gives

$$\begin{bmatrix} V_q^T M^\nu V_q - I & 0 & V_q^T \mathbf{F}_\lambda^\nu \\ V_p^T M^\nu V_q & V_p^T M^\nu V_p - I & V_p^T \mathbf{F}_\lambda^\nu \\ n_{\mathbf{u}}^\nu V_q & n_{\mathbf{u}}^\nu V_p & n_\lambda^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{q}}^\nu \\ \Delta\bar{\mathbf{p}}^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} V_q^T \mathbf{r}^\nu \\ V_p^T \mathbf{r}^\nu \\ n^\nu \end{bmatrix}. \tag{3.25}$$

As in our analysis before, we have used the facts that $\mathcal{P}$ is an invariant subspace of $\mathbf{F}_\mathbf{u}$ and that $V_q^T V_p = V_p^T V_q = 0$ since $\mathcal{P}$ and $\mathcal{Q}$ are orthogonal subspaces.

The method used to solve equation (3.25) is similar to that for solving (3.12), except that solving the $\mathcal{Q}$-equations requires two sets of Picard iterations rather than one. The $\mathcal{Q}$-equations here are

$$(V_q^T M^\nu V_q - I)\Delta\bar{\mathbf{q}}^\nu + V_p^T \mathbf{F}_\lambda^\nu \Delta\lambda^\nu = -V_q^T \mathbf{r}^\nu,$$

which implies that

$$\Delta\bar{\mathbf{q}}^\nu = -(V_q^T M^\nu V_q - I)^{-1} V_q^T \mathbf{r}^\nu - (V_q^T M^\nu V_q - I)^{-1} V_q^T \mathbf{F}_\lambda^\nu \Delta\lambda^\nu.$$

Let $\Delta\bar{\mathbf{q}}_r^\nu = -(V_q^T M^\nu V_q - I)^{-1} V_q^T \mathbf{r}^\nu$ and $\Delta\bar{\mathbf{q}}_\lambda^\nu = -(V_q^T M^\nu V_q - I)^{-1} V_q^T \mathbf{F}_\lambda^\nu$, so that $\Delta\bar{\mathbf{q}}^\nu = \Delta\bar{\mathbf{q}}_r^\nu + \Delta\bar{\mathbf{q}}_\lambda^\nu \Delta\lambda^\nu$. Instead of actually forming the matrix inverse

$$(V_q^T M^\nu V_q - I)^{-1},$$

the inverse is approximated by using the first ℓ terms of its convergent Neumann series. This amounts to applying the Picard iteration scheme as described in Section 3.3.1. The term $\Delta\bar{\mathbf{q}}_r^\nu$ is found exactly as before, while $\Delta\bar{\mathbf{q}}_\lambda^\nu$ can be found by Picard iteration with $V_q^T \mathbf{F}_\lambda^\nu$ in place of $V_q^T \mathbf{r}^\nu$. Specifically, writing $\Delta\mathbf{q}_r^\nu = V_q \Delta\bar{\mathbf{q}}_r^\nu$ and $\Delta\mathbf{q}_\lambda^\nu = V_q \Delta\bar{\mathbf{q}}_\lambda^\nu$ to utilize vectors in $\mathbb{R}^N$, these iterations are

$$\begin{aligned} &\Delta\mathbf{q}_r^{[0]} = \mathbf{0}, \\ &\Delta\mathbf{q}_r^{[k]} = QM\Delta\mathbf{q}_r^{[k-1]} + Q\mathbf{r}, \quad k = 1, \ldots, \ell, \\ &\Delta\mathbf{q}_r^\nu = \Delta\mathbf{q}_r^{[\ell]}, \end{aligned} \tag{3.26}$$

and

$$\begin{aligned}
\Delta\mathbf{q}_\lambda^{[0]} &= \mathbf{0}, \\
\Delta\mathbf{q}_\lambda^{[k]} &= QM\Delta\mathbf{q}_\lambda^{[k-1]} + Q\mathbf{F}_\lambda, \quad k = 1, \ldots, \ell, \\
\Delta\mathbf{q}_\lambda^{\nu} &= \Delta\mathbf{q}_\lambda^{[\ell]}.
\end{aligned} \tag{3.27}$$

As before, we use the fact that $Q = I - V_p V_p^T$ along with the modified Gram-Schmidt method to perform the projections onto $\mathcal{Q}$, rather than constructing the large matrix Q explicitly.

Note that the equations defining $\Delta\bar{\mathbf{q}}_r^\nu$ and $\Delta\bar{\mathbf{q}}_r^\nu$ involve the same matrix $V_q^T M^\nu V_q - I$ but different right-hand sides. If we were using, for example, an LU factorization of the matrix, we could take advantage of this fact to reduce the total number of computations required to solve both systems. Unfortunately this is not possible in our situation since we are using an iterative method to solve the systems. The additional expense of solving a second system, which can be considerable, is a significant drawback of utilizing the Newton-Picard method within an arclength continuation context.

Computing $\Delta\bar{\mathbf{q}}_r^\nu$ and $\Delta\bar{\mathbf{q}}_\lambda^\nu$ is not sufficient to determine $\Delta\bar{\mathbf{q}}^\nu$, since the term $\Delta\lambda^\nu$ is still unknown. Replacing $\Delta\bar{\mathbf{q}}^\nu$ with $\Delta\bar{\mathbf{q}}_r^\nu + \Delta\bar{\mathbf{q}}_\lambda^\nu\Delta\lambda^\nu$ in (3.25) and rearranging the terms from the second and third block rows of the Jacobian matrix yields the $\mathcal{P}$ equations

$$\begin{bmatrix} V_p^T M^\nu V_p - I & V_p^T(\mathbf{F}_\lambda^\nu + M^\nu\Delta\mathbf{q}_\lambda^\nu) \\ n_{\mathbf{u}}^\nu V_p & n_\lambda^\nu + n_{\mathbf{u}}^\nu\Delta\mathbf{q}_\lambda^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T(\mathbf{r}^\nu + M^\nu\Delta\mathbf{q}_r^\nu) \\ n^\nu + n_{\mathbf{u}}^\nu\Delta\mathbf{q}_r^\nu \end{bmatrix}. \tag{3.28}$$

Thus, the procedure to solve system (3.25) is as follows. First, compute $\Delta\mathbf{q}_r^\nu$ and $\Delta\mathbf{q}_\lambda^\nu$ by the Picard iterations (3.26) and (3.27), respectively. Next, solve the $\mathcal{P}$-equations (3.28) for $\Delta\bar{\mathbf{p}}^\nu$ and $\Delta\lambda^\nu$, and then set $\Delta\mathbf{q}^\nu = \Delta\mathbf{q}_r^\nu + \Delta\mathbf{q}_\lambda^\nu\Delta\lambda^\nu$. Finally, set

$$\begin{aligned}
\mathbf{u}^{\nu+1} &= \mathbf{u}^\nu + V_p\Delta\bar{\mathbf{p}}^\nu + \Delta\mathbf{q}^\nu, \\
\lambda^{\nu+1} &= \lambda^\nu + \Delta\lambda^\nu.
\end{aligned}$$

3.4.2 Time-Periodic Solutions

The equations of Section 3.3.3 can be modified in a similar manner to allow for the introduction of a parameterization equation for computing turning points on branches of time-periodic solutions. In this section we briefly outline the resulting equations to be solved by the Newton Picard method.

As above, let s denote the new parameter, defined by the parameterization equation $n(\mathbf{u}_0, T, \lambda, s) = 0$. We view the initial point $\mathbf{u}_0$, the period T, and the parameter λ as functions of s. For each s, the system of equations to be solved by Newton's method in the time-periodic case is

$$\begin{bmatrix} \boldsymbol{\phi}(\mathbf{u}_0(s), T(s), \lambda(s)) - \mathbf{u}_0(s) \\ \psi(\mathbf{u}_0(s), T(s), \lambda(s)) \\ n(\mathbf{u}_0(s), T(s), \lambda(s), s) \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

Let $\mathbf{r}^\nu = \boldsymbol{\phi}(\mathbf{u}_0^\nu, T^\nu, \lambda^\nu) - \mathbf{u}_0^\nu$. At each Newton iteration, we must solve the system

$$\begin{bmatrix} M^\nu - I & \boldsymbol{\phi}_T^\nu & \boldsymbol{\phi}_\lambda^\nu \\ \psi_{\mathbf{u}}^\nu & \psi_T^\nu & \psi_\lambda^\nu \\ n_{\mathbf{u}}^\nu & n_T^\nu & n_\lambda^\nu \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}_0^\nu \\ \Delta T^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} \mathbf{r}^\nu \\ \psi^\nu \\ n^\nu \end{bmatrix}, \tag{3.30}$$

for $\Delta\mathbf{u}_0^\nu$, ΔT^ν and $\Delta\lambda^\nu$, and then set

$$\begin{aligned} \mathbf{u}_0^{\nu+1} &= \mathbf{u}_0^\nu + \Delta\mathbf{u}_0^\nu, \\ T^{\nu+1} &= T^\nu + \Delta T^\nu, \\ \lambda^{\nu+1} &= \lambda^\nu + \Delta\lambda^\nu. \end{aligned} \tag{3.31}$$

Using the orthogonal decomposition $\mathbb{R}^N = \mathcal{P} \oplus \mathcal{Q}$ and bases V_p and V_q for the subspaces $\mathcal{P}$ and $\mathcal{Q}$, we can decompose system (3.30) to get

$$\begin{bmatrix} V_q^T M^\nu V_q - I & 0 & 0 & V_q^T \boldsymbol{\phi}_\lambda^\nu \\ V_p^T M^\nu V_q & V_p^T M^\nu V_p - I & V_p^T \boldsymbol{\phi}_T^\nu & V_p^T \boldsymbol{\phi}_\lambda^\nu \\ \psi_{\mathbf{u}}^\nu V_q & \psi_{\mathbf{u}}^\nu V_p & \psi_T^\nu & \psi_\lambda^\nu \\ n_{\mathbf{u}}^\nu V_q & n_{\mathbf{u}}^\nu V_p & n_T^\nu & n_\lambda^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{q}}^\nu \\ \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} V_q^T \mathbf{r}^\nu \\ V_p^T \mathbf{r}^\nu \\ \psi^\nu \\ n^\nu \end{bmatrix}. \tag{3.32}$$

To obtain this system, we have also used the simplification that $V_q^T \boldsymbol{\phi}_T^\nu \approx 0$, as we did in Section 3.3.3. By reading the first block equation of (3.32), we obtain the $\mathcal{Q}$-equations. These equations are identical to the equations in the equilibrium solution

case, and hence can be solved in the same way: apply Picard iteration to obtain $\Delta\mathbf{q}_r^\nu$ and $\Delta\mathbf{q}_\lambda^\nu$. Substituting these into (3.32) gives the $\mathcal{P}$-equations

$$\begin{bmatrix} V_p^T M^\nu V_p - I & V_p^T \phi_T^\nu & V_p^T(\phi_\lambda^\nu + M^\nu \Delta\mathbf{q}_\lambda^\nu) \\ \psi_{\mathbf{u}}^\nu V_p & \psi_T^\nu & \psi_\lambda^\nu + \psi_{\mathbf{u}}^\nu \Delta\mathbf{q}_\lambda^\nu \\ n_{\mathbf{u}}^\nu V_p & n_T^\nu & n_\lambda^\nu + n_{\mathbf{u}}^\nu \Delta\mathbf{q}_\lambda^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \\ \Delta\lambda^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T(\mathbf{r}^\nu + M^\nu \Delta\mathbf{q}_r^\nu) \\ \psi^\nu + \psi_{\mathbf{u}}^\nu \Delta\mathbf{q}_r^\nu \\ n^\nu + n_{\mathbf{u}}^\nu \Delta\mathbf{q}_r^\nu \end{bmatrix},$$

which are solved by direct methods since this is a small-dimensional system. In the end, we obtain

$$\mathbf{u}_0^{\nu+1} = \mathbf{u}_0^\nu + V_p \Delta\bar{\mathbf{p}}^\nu + \Delta\mathbf{q}_r^\nu + \Delta\mathbf{q}_\lambda^\nu \Delta\lambda^\nu$$

and $T^{\nu+1}$ and $\lambda^{\nu+1}$ as in (3.31).

3.5 Applications

Many applications of the recursive projection method and the Newton-Picard method have appeared in the literature since these ideas were first introduced. P. Love, one of Keller's former Ph. D. students at the California Institute of Technology, used the RPM as an aid for solving the Navier-Stokes equations [84]. His goal was to do a bifurcation study of Kolmogorov flow and flow in the Taylor-Couette problem. However, Love did not use the RPM to wrap around existing time-stepper codes for those flows as we have done here. Instead, he formulated the Navier-Stokes equations in such a way that Newton's method could be applied directly to compute their solution. At each Newton step, a linear system of equations must be solved. Love solves this system using an iterative method obtained from the numerical integration of an ODE, and then applies the RPM on this inner iteration to stabilize it and speed convergence. In this regard he was successful. His RPM-Navier-Stokes solver was able to detect bifurcation points and was able to follow some unstable solution branches. It should be emphasized, however, that Love's original solution algorithm was designed with the intention of ultimately using the RPM.

K. Lust and his colleagues at K. U. Leuven have further applied the Newton-Picard method to such problems as the bifurcation analysis of periodic solutions of

delay differential equations [90] and the computation of period-doubling bifurcations [50]. Ito and Kumamoto [64] used the Newton-Picard method to focus on the computation of fold bifurcation points in nonautonomous systems of ODEs. An application of the Newton-Picard method to perform a bifurcation analysis of the flow in a driven cavity can be found in [137], which used the software package PDECONT. In [141], the Newton-Picard method is used to compute periodic states of cyclically operated chemical processes. Atmospheric scientists have even used the method, applying it to computing baroclinic instabilities [115]. Finally, we mention I. G. Kevrekidis, who, along with many others, has produced numerous papers on applying the RPM and the Newton-Picard method to perform a "coarse" stability and bifurcation analysis using only microscopic-level chemical process codes. His applications enable a bifurcation analysis to be performed even when the governing macroscopic (differential) equations are not known. The intrepid reader may consult [54, 75, 101, 112, 134, 79, 91, 77] for details of his work.

3.6 Summary

In this chapter, we have discussed a class of general solution procedures that we have collectively termed Newton-Picard methods. These procedures, although differing in fine details, all share the same basic idea: split the solution of a mathematical problem into two pieces, one corresponding to the part of the problem representing weakly convergent or unstable modes, the other corresponding to the complement. We have looked at two of these methods in detail: the recursive projection method of Shroff and Keller [121] and the Newton-Picard method of Lust [85], and we discussed how the latter is a generalization of the former.

We have given only enough details of the RPM and the Newton-Picard method so that we can understand these methods sufficiently to apply them to our goal: creating a time-stepper based bifurcation code. Much more information on these methods, including convergence proofs, simple applications, and further refinements,

can be found in the original works [121], [74], and [85]. Although both Lust and Keller mention that their methods can be applied as a wrap-around for an existing time-stepper code, neither actually attempts it. The next two chapters describe how we were able to accomplish this for the Taylor-Couette problem.

Chapter 4

The Taylor-Couette Problem

In this chapter, we more fully describe the Taylor-Couette problem, including a brief history of the problem and a discussion of various studies that have been performed. We will also give details of the numerical scheme used by Adair [2] and some of the dynamics and bifurcations he observed with his program. The review article by Donnelly [47] gives a more complete historical perspective of the problem.

4.1 Description and History of the Problem

The Taylor-Couette problem has been studied extensively, in various forms, for hundreds of years. In fact, its origins date back to Newton [47]. The basic experimental apparatus consists of a viscous incompressible fluid placed in the annular region between two concentric coaxial circular cylinders. The experiment itself involves studying the flow patterns that result as one or both of the cylinders are rotated. A variety of apparatus configurations appear in the literature: the ends (top and bottom) of the apparatus can be held fixed, rotated with one of the cylinders, or the top can be removed altogether. Often only the inner cylinder is rotated while the outer cylinder is kept fixed, but experiments have been performed where both cylinders are rotated in the same direction (co-rotation) or in opposite directions (counter-rotation). See [48] for a comprehensive discussion of various apparatuses that have been used in experiments.

The Taylor-Couette problem has been so widely studied because it is conceptually simple to understand but exhibits a large variety of complex fluid flow behavior. In particular, the flow patterns observed can change quite drastically and dramatically with changes in the rotation rates of one or both of the cylinders. Mathematically, such changes are driven by the nonlinear terms found in the governing fluid flow equations, the Navier-Stokes equations. It has been the goal of many experimental physicists, engineers, and mathematicians to more fully understand this nonlinear behavior and to try to explain it. Physical experiments, mathematical analysis, and numerical simulations are the tools used to study this problem and to study the character of the fluid flows produced.

The problem is partly named after M. Maurice Couette, who in the late 19th century built a coaxial cylinder apparatus as a means of measuring fluid viscosity [35]. In his apparatus, the inner cylinder was suspended from a torsion fiber with the outer cylinder rotating. Viscosity was computed by measuring the torque on the inner cylinder. One of the flows that Couette observed with his apparatus now bears his name. Couette flow is a steady laminar flow which has a purely azimuthal component and is dependent only on the radial position. However, Couette flow is an exact solution of the Navier-Stokes equations only in the case where the cylinders have infinite length. Couette flow only is observed in physical experiments near the middle of very long cylinders.

The name of G. I. Taylor was attached to the problem following a groundbreaking paper in 1923 that summarized the results of an extensive study he conducted on the problem [132]. Taylor constructed an experimental apparatus and compared the flow patterns he observed with the linear stability theory for infinite cylinders he had developed. Theory and experiment agreed nicely, he concluded. Taylor was also among the first to do a detailed linear stability analysis of the problem. In doing so, he calculated the critical rotation rate at which Couette flow loses stability, as well as the axial wavelength of the disturbed flow. With a fixed outer cylinder, as

the inner cylindrical rotation rate is increased, Couette flow becomes unstable and a new flow becomes the stable solution. This new flow is a steady axisymmetric flow that consists of a number of toroidal regions encircling the inner cylinder in which the fluid circulates. These toroidal vortices are now known as Taylor vortices and the associated flow is now called Taylor vortex flow.

Taylor's theory and linear stability analysis assume that the cylinders have infinite length; in this way, end effects may be ignored. Of course, in any physical experiment, end effects will be present and will need to be dealt with. Taylor did this by using 90 cm long cylinders in which the fluid between the cylinders was confined to a gap of less than one centimeter. He analyzed the flow far away from either end, near the middle of his apparatus. Since then, other techniques have been developed to simulate infinitely long cylinders in experiments. The use of "ramps," where the outer cylinder is gradually tapered inward [3], currently is one of the most advanced ways to imitate infinitely long cylinders in experimental apparatuses [97].

Many other experiments have used various approaches to minimize end effects so that experimental results may be compared with the infinite cylinder theory. Notable among these is the work of Coles [34], whose experiments confirmed another consequence of the nonlinear terms of the Navier-Stokes equations: non-uniqueness of solutions. In fact, Coles observed as many as 20 to 25 different flows for the same problem parameters (the speed of the cylinders' rotation and the height and radii of the cylinders). These various flows were distinguished by different numbers of Taylor vortices observed and different numbers of waves appearing in a time-dependent flow called wavy vortex flow. Which flow was observed depended on how the state of the apparatus was achieved; for example, slowly increasing the rotation rate versus suddenly doing so. The experiments of Benjamin and Mullin [12, 13] and Burkhalter and Koschmieder [18, 19] also have exhibited this non-uniqueness of flow solutions.

The short cylinder Taylor-Couette problem is also of interest, and there have been experimental studies to analyze end effects in short cylinder arrangements. Among

the first of these was Cole [33], who showed that the appearance of wavy vortex flows heavily depends on cylindrical height. T. B. Benjamin also considered end effects in his work. Not only did he perform physical experiments with relatively short cylinders [11], but he also developed the theory, based on Leray-Schauder degree theory, to mathematically analyze the effects of the ends [10]. In his experiments, Benjamin also observed flows that seemed to contradict the generally held belief that the fluid flow should be directed inward at the ends due to the decrease of the centrifugal force near the ends caused by the no-slip boundary conditions there [108]. These flows are called "anomalous modes," and cannot be achieved by gradually changing the cylindrical velocity [104]. Further experimental, analytical, and numerical work on these anomalous modes (and related modes, such as single-cell Taylor vortex flow) can be found in the papers of Benjamin and Mullin [12], Lorenzen and Mullin [83], Cliffe, Kobine, and Mullin [29], Cliffe and Mullin [31], Anson, Mullin, and Cliffe [8], Cliffe [26], and Bolstad and Keller [16]. Finally, we note that the experimental and numerical work of Tavener, Mullin, and Cliffe [131] clearly shows the effect of the ends by considering a variant of the traditional Taylor-Couette problem, where the ends of the apparatus rotate with the inner cylinder.

Other than Taylor vortex flow, many other types of flows have been observed since Taylor's ground-breaking work. For instance, the wavy vortex flow mentioned above is observed in experiments with a stationary outer cylinder as the inner cylinder's rotation rate is increased to the point where the Taylor vortex flow loses stability. This wavy vortex flow is similar to Taylor vortex flow, but it is a time-dependent flow in which the vortices move with a well-defined frequency; one can think of this flow as superimposing a travelling azimuthal wave upon the Taylor vortices. A doubly periodic (quasi-periodic) flow is observed when the rotation rate becomes even faster [56]. An extensive series of experiments was performed by Andereck, Swinney, and others [5, 6], in which an even wider variety of flows was observed, with names such as modulated vortex flow, spiral flow, interpenetrating spirals, twisted Taylor vortices,

and braided vortices. Turbulent flows, with no apparent large-scale features, have also been observed at sufficiently large rotation rates [6, 53]. A survey of many of these different flows can be found in [39], where the authors also include details of the theory behind them.

We should note that many, but not all, of the experiments mentioned above were performed with the typical setup of the outer cylinder at rest and inner cylinder rotating. Additionally, experiments have been performed where both cylinders rotate. Both co-rotation (cylinders rotating in the same direction) and counter-rotation (cylinders rotating in the opposite direction) were studied by Taylor. Co-rotation has also been studied by Andereck, Dickman, and Swinney [5] and by Coles [34], while counter-rotating experiments have been performed Synder [122, 123], Andereck, Liu, and Swinney [6], Schulz and Pfister [118], and also by Coles [34]. For our numerical bifurcation results outlined in Chapter 6, we will be focusing on the counter-rotating problem.

In addition to physical experiments, the Taylor-Couette problem has been extensively studied using analytical methods. Among the first to do this was Taylor, but many others have followed in his footsteps. Typically, simplifying assumptions are made to make the analysis easier. As we've already mentioned, infinitely long cylinders are often assumed so that end effects and axial boundary conditions can be ignored. This assumption is also frequently made because the analytical formulas for Couette flow are an exact solution of the Navier-Stokes equations only for infinitely long cylinders, not for finite-length cylinders. Other simplifying assumptions frequently made are that the flow is axisymmetric, the flow is periodic in the axial direction, or the flow is time independent. These assumptions restrict the types of flows that can be analyzed mathematically. As a consequence, numerical experiments have become increasingly important in understanding the Taylor-Couette problem more fully. Since our work is numerical, we will not discuss analytical methods in this dissertation. The interested reader may consult [25] or Chapter VII of [23] and

the references therein. End effects are treated analytically by Schaeffer [116] and Benjamin [10].

A considerable number of numerical investigations of the Taylor-Couette problem can be found in the literature for both simulating the Navier-Stokes equations and for detecting and studying changes in the flow patterns as the cylindrical velocities are varied. Among the first of these was the numerical work of Meyer in the 1960's. In [95], he studied Taylor vortex flow by using a streamfunction-vorticity formulation of the time-dependent axisymmetric Navier Stokes equations. Meyer assumed periodic axial boundary conditions to eliminate end effects. Similar numerical methods were later employed by Alziary de Roquefort and Grillaud [4] and by Neitzel [107], but they included end effects in their computations. In later work, Meyer [96] extended his method to allow for the computation of nonaxisymmetric flows, notably wavy vortex flows, by expanding the velocities as a Fourier series in the angular variable θ.

Much numerical work has been performed by Cliffe and others [32, 31, 30, 27, 108, 8], who computed steady axisymmetric flows in very short cylinder arrangements using a finite element method. Their bifurcation results use the numerical bifurcation methods alluded to in Chapter 1. H. B. Keller and co-workers have developed multigrid-based algorithms for the Taylor-Couette problem [16, 40]. They restricted their computations to steady axisymmetric solutions, but they did consider finite-length cylinders and hence had to take into account end effects. Also among the numerical experimenters is R. Meyer-Spasche, who, with others, developed the code called TAYPERIO, which also computes steady axisymmetric solutions of the Taylor-Couette problem [97, 98, 99, 100, 125]. TAYPERIO uses a combination of finite differences in the radial variable with Fourier series expansions in the axial variable in such a way as to force periodic boundary conditions, so end effects can be ignored, and to force a mid-plane symmetry of the solution.

To more fully understand the Taylor-Couette problem, we must be able to compute more than just steady axisymmetric solutions. Even the rather simple wavy

vortex flow is a time-dependent nonaxisymmetric flow. Therefore, much work has been done with the goal of computing such solutions, of which we mention only a few. Keller and Schröder [117] expanded Keller's original multigrid-based algorithm mentioned above to allow for the computation of time-dependent and nonaxisymmetric solutions. In particular, they were looking for travelling wave solutions (of which wavy vortex flow is one), that are steady flows when viewed in a rotating reference frame. The numerical code of Marcus [93] is also able to compute wavy vortex flows [94] and modulated (quasi-periodic) vortex flows [36]. Wavy vortex flows have also been studied numerically by Moser, Moin, and Leonard [103] and by Edwards, Tuckerman, Friesner, and Sorensen [49]. The latter work is of interest because of their use of Krylov subspace methods in the solution algorithm. The more recent results of Hoffmann and Lücke [62] were obtained from a code based on finite differences and Fourier series expansions to analyze spiral vortex flows.

Finally, we mention the numerical work of Streett and Hussaini [128]. They used a multi-step scheme to solve the time-dependent nonaxisymmetric Navier-Stokes equations, and they focused on short cylinder lengths. Their numerical experiments are the closest to that of Adair [2], which we now describe in more detail.

4.2 Numerical Solution Method of Adair

In this section, we briefly describe the numerical method used by Adair [2] to solve the finite-length Taylor-Couette problem. Adair gave his completed computer program the name TAY, and we continue his nomenclature. Later in this section we give a description of the computational results obtained by Adair as well as some later results of J. W. Thomas using the TAY code [1]. We conclude the section by discussing aspects of the observed solution bifurcations.

First, we introduce some notation. Let r_1 and r_2 denote the inner and outer cylinder radii, respectively, and let $\eta = r_1/r_2$, the radius ratio. We define the gap

width to be $d = r_2 - r_1$. Let h denote the height of the cylinders, and let Ω_1 and Ω_2 denote the angular velocity of the inner and outer cylinders, respectively.

The equations that describe the flow of a viscous incompressible fluid lying between concentric cylinders in the Taylor-Couette problem are the Navier-Stokes equations. In the cylindrical geometry of the Taylor-Couette problem, it is natural to express these equations using cylindrical coordinates with spatial variables r, θ, and z. The equations are put into non-dimensional form by choosing the characteristic length to be the gap width d and the characteristic velocity to be the velocity of the inner cylinder $r_1\Omega_1$. To this non-dimensionalization there correspond two Reynolds numbers, one for each cylinder. We define the inner and outer Reynolds numbers to be

$$R_i = \frac{r_1\Omega_1 d}{\nu} \qquad \text{and} \qquad R_o = \frac{r_2\Omega_2 d}{\nu},$$

respectively, where ν denotes the kinematic viscosity of the fluid. These quantities are the non-dimensionalized velocities of the inner and outer cylinders, and hence cylindrical rotation rates can be controlled by changing the appropriate Reynolds number. The geometry of the problem can be described entirely in terms of the non-dimensional radius ratio η and the aspect ratio $\Gamma = h/d$.

In the TAY code, $r = 0$ corresponds to the central axis of the Taylor-Couette apparatus, $z = 0$ corresponds to the bottom of the cylinders, and $z = \Gamma$ corresponds to the top of the cylinders. The ends of the cylinders are fixed, and no-slip velocity boundary conditions are imposed at the ends and at the cylindrical walls. Pressure boundary conditions are treated similarly to those given in [24], but the spatial discretization has been slightly changed. In cylindrical coordinates, with these non-dimensional variables and parameters, the vector form of the Navier-Stokes equations is

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} &= \nabla^2\mathbf{v} + L\mathbf{v} - \nabla p - R_i[(\mathbf{v}\cdot\nabla)\mathbf{v} + A(\mathbf{v})], \\ \nabla\cdot\mathbf{v} &= 0, \end{aligned} \tag{4.1}$$

where $\mathbf{v} = (u, v, w)$ is the velocity field and u, v, and w, are the radial, azimuthal, and axial velocity components, respectively, and p is the pressure. The operators which appear in (4.1) are defined as:

$$L\mathbf{v} = \left(-\frac{u + 2\frac{\partial v}{\partial \theta}}{r^2}, -\frac{v - 2\frac{\partial u}{\partial \theta}}{r^2}, 0 \right), \quad A(\mathbf{v}) = \left(-\frac{v^2}{r}, \frac{uv}{r}, 0 \right),$$

$$\nabla p = \left(\frac{\partial p}{\partial r}, \frac{1}{r}\frac{\partial p}{\partial \theta}, \frac{\partial p}{\partial z} \right),$$

$$\mathbf{v} \cdot \nabla = u\frac{\partial}{\partial r} + \frac{v}{r}\frac{\partial}{\partial \theta} + w\frac{\partial}{\partial z},$$

$$\nabla \cdot \mathbf{v} = \frac{1}{r}\frac{\partial (ru)}{\partial r} + \frac{1}{r}\frac{\partial v}{\partial \theta} + \frac{\partial w}{\partial z},$$

and

$$\nabla^2 = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r} \right) + \frac{1}{r^2}\frac{\partial^2}{\partial \theta^2} + \frac{\partial^2}{\partial z^2}.$$

To utilize the azimuthal periodicity that must be present in this annular region, the solution velocity and pressure are expanded as a truncated Fourier series with respect to θ. Thus, we have the approximations

$$u(r, \theta, z, t) \approx \frac{1}{2}u_{10}(r, z, t) + \sum_{\ell=1}^{N} \left(u_{1\ell}(r, z, t)\cos(\ell\theta) + u_{2\ell}(r, z, t)\sin(\ell\theta) \right)$$

$$v(r, \theta, z, t) \approx \frac{1}{2}v_{10}(r, z, t) + \sum_{\ell=1}^{N} \left(v_{1\ell}(r, z, t)\cos(\ell\theta) + v_{2\ell}(r, z, t)\sin(\ell\theta) \right)$$

$$w(r, \theta, z, t) \approx \frac{1}{2}w_{10}(r, z, t) + \sum_{\ell=1}^{N} \left(w_{1\ell}(r, z, t)\cos(\ell\theta) + w_{2\ell}(r, z, t)\sin(\ell\theta) \right)$$

$$p(r, \theta, z, t) \approx \frac{1}{2}p_{10}(r, z, t) + \sum_{\ell=1}^{N} \left(p_{1\ell}(r, z, t)\cos(\ell\theta) + p_{2\ell}(r, z, t)\sin(\ell\theta) \right)$$

A spatial discretization is then used in the rz-plane for the corresponding Fourier coefficients. It should be noted that this approach of using finite differences combined with a Fourier expansion has been used before to study the Taylor-Couette problem [96, 62]. Adair uses centered differences for all second derivatives, but a variant of the Lax-Wendroff difference scheme is used for first derivatives. Rather than treating each Fourier coefficient as a function of two spatial variables and using the two-dimensional

Lax-Wendroff scheme for spatial first derivatives, each spatial first derivative is instead discretized by the one-dimensional Lax-Wendroff scheme independently. The resulting scheme is not the true two-dimensional Lax-Wendroff scheme, but nevertheless works well. See page 246 of [135] for the exact form of this scheme.

The projection method of Chorin [24] and Témam [133] is used to solve for these unknown coefficients, the same method used by Griebel, Dornseifer, and Neunhoeffer [59]. This method splits the problem so that it is solved in three steps. First, an explicit Euler (forward-time) method is used to approximate the temporal derivative:

$$\frac{d\mathbf{v}}{dt} \approx \frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t},$$

where n represents the time step number. For notational convenience, we define

$$F(\mathbf{v}) = \nabla^2 \mathbf{v} + L\mathbf{v} - R_i[(\mathbf{v} \cdot \nabla)\mathbf{v} + A(\mathbf{v})].$$

The Chorin-Témam scheme evaluates the pressure at the $(n+1)$st time step, which gives us the following discrete form of the Navier-Stokes equations:

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t F(\mathbf{v}^n) - \Delta t \nabla p^{n+1}. \tag{4.2}$$

If we define an auxiliary vector field $\mathbf{v}^*$ by

$$\mathbf{v}^* = \mathbf{v}^n + \Delta t F(\mathbf{v}^n), \tag{4.3}$$

then equation (4.2) becomes

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t \nabla p^{n+1}. \tag{4.4}$$

Next, applying the discrete divergence operator to (4.4), and utilizing the incompressibility of the fluid so that $\nabla \cdot \mathbf{v}^{n+1} = 0$, we get the Poisson equation

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}^*, \tag{4.5}$$

which is then solved by the Gauss-Seidel iterative method for p^{n+1}. This pressure p^{n+1} and the auxiliary field $\mathbf{v}^*$ are then combined as indicated in (4.4) to obtain the

velocity $\mathbf{v}^{n+1}$ at the next time step. In summary, equations (4.3), (4.5), and (4.4), taken in that order, form the time-stepper code written by Adair.

It is worth remarking at this point that the Chorin-Témam projection scheme can be viewed as a function that maps the velocity $\mathbf{v}^n$ to the velocity $\mathbf{v}^{n+1}$ at the next time step. The pressure term is used only as an intermediate value to compute $\mathbf{v}^{n+1}$. Put another way, the velocity $\mathbf{v}^{n+1}$ is dependent only upon $\mathbf{v}^n$, since the pressure term p^{n+1} is also dependent only upon $\mathbf{v}^n$. This view of the Chorin-Témam scheme will play an important role in the construction of our wrap-around code, which we will describe in more detail in Chapter 5.

As discussed in the last section, many numerical experiments in the literature assume *a priori* that the flow is axisymmetric (independent of θ), or that the flow is periodic in the axial direction, or even that the flow is steady. These assumptions are not made by Adair, allowing for more complicated flow patterns. Time-dependent solutions can be computed, as well as non-axisymmetric flows. End effects also are taken into account via no-slip boundary conditions at the top and bottom, which are assumed to be stationary. The TAY code is better suited for use in a short cylinder system because there are fewer multiple solutions (for given Reynolds numbers) in short cylinders and because the computational expense required for long cylinders is quite large.

4.2.1 Numerical Experiments Conducted with TAY

In this section, we describe some of the numerical results obtained from the TAY code as detailed in [2], focusing on the axisymmetric flows observed there. The basic process used in the numerical experiments is quite close to the process actually used in physical experiments. Physical experiments begin by slowly rotating one of the cylinders. The system is then given some time to settle and to develop a well-defined flow pattern. Next, the rotation rate of one of the cylinders is changed slightly, and the system is allowed to settle again. This process is repeated until the desired combination of inner and outer cylindrical rotation rates is obtained.

In the numerical experiments of Adair, the TAY code is started with a very slowly rotating inner cylinder and stationary outer cylinder. Couette flow is used as an initial condition. Like the physical experiments, this numerical system is allowed to equilibrate, which for a numerical experiment means that many time steps are used in computing the solution. Then the rotation rate of one of the cylinders is changed slightly and again the system is allowed to sit for many time steps. This process uses the flow solution at the previous rotation rate as an initial condition for finding the solution at the new rotation rate, which is precisely what happens in physical experiments.

Recall that R_i and R_o denote the inner and outer cylinder Reynolds numbers, respectively. The rotation rates of the two cylinders are controlled by changing these parameters. In the experiments of Adair, an aspect ratio of $\Gamma = 2.4$ and a radius ratio of $\eta = 0.615$ were studied. The numerical experiments were begun by setting $R_i = 1$ and $R_o = 0$, using Couette flow as an initial condition, and allowing the system to equilibrate. The resulting flow pattern is a very weak axisymmetric two-cell Taylor vortex flow that is symmetric about the midplane of the cylinders ($z = \Gamma/2 = 1.2$). Figure 4.1 shows a representative cross-section of such a two-cell flow.

The value of R_i is then increased in small increments to $R_i = 300$, again allowing sufficient time to reach equilibrium. The flows along this path are still two-cell axisymmetric Taylor vortex flows and are qualitatively similar to the flow in figure 4.1. Next, the outer Reynolds number R_o is slowly changed from 0 to -90, so we now are considering a counter-rotating Taylor-Couette problem. As R_o is changed in this manner, the flow remains a two-cell Taylor vortex flow, but the cells begin to show more pronounced changes in character. In particular, they become more compressed in the axial direction as R_o is decreased.

However, as R_o is decreased to approximately -90, the observed flow behavior changes significantly. A time-dependent flow appears, destroying the midplane symmetry enjoyed by the previous two-cell flow. Two axisymmetric Taylor vortices

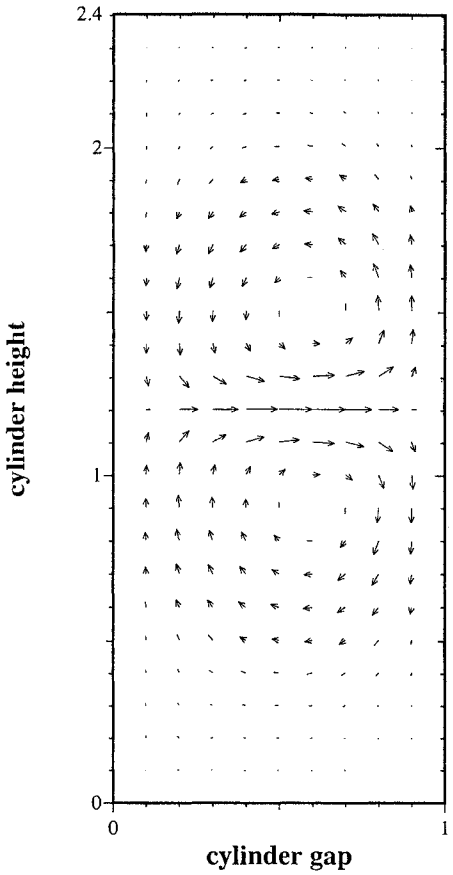

Figure 4.1: *Radial-axial velocity field of representative two-cell Taylor vortex flows projected onto a vertical cylinder slice.*

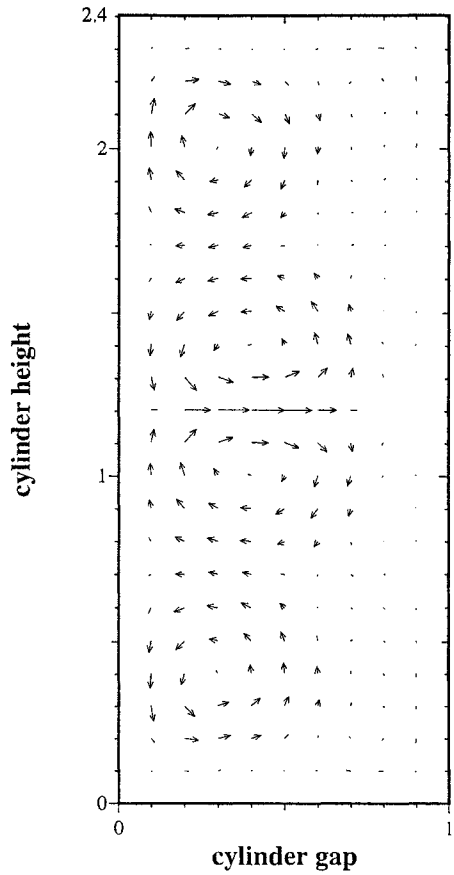

Figure 4.2: *Radial-axial velocity field of representative four-cell Taylor vortex flows projected onto a vertical cylinder slice.*

remain, but the interface between them oscillates. As R_o is decreased further, even more complicated flow behavior arises. The flow becomes what Adair termed "disorganized Taylor vortex flow." The number, size, shape, and location of the vortices change in a time-dependent fashion for fixed R_o. At certain times, there are two cells; at others, four; and at still others, three cells can be observed. Based on preliminary calculations, it appears that this disorganized behavior is also time-periodic. Surprisingly, all such disorganized flows are still axisymmetric.

The disorganized Taylor vortex flow continues as R_o is further decreased until $R_o \approx -320$ is reached, where there is an axisymmetric four-cell Taylor vortex flow.

From (R_i, R_o)	*To* (R_i, R_o)	*Observed Flow*
$(1, 0)$	$(300, 0)$	two-cell steady Taylor vortex flow
$(300, 0)$	$(300, -90)$	two-cell steady Taylor vortex flow
$(300, -90)$	$(300, -300)$	time-dependent flow
$(300, -320)$	$(300, -320)$	four-cell steady Taylor vortex flow

Table 4.1: *Summary of axisymmetric flows observed by R. Adair [2].*

Figure 4.2 shows a typical cross-section of a four-cell Taylor flow. This flow appears to be steady; the time periodicity of the disorganized flow has been lost. Furthermore, the z-symmetry about the midplane has been re-established for this four-cell flow. We have summarized these axisymmetric flows in table 4.1.

Further numerical experiments of axisymmetric flows recently have been made by J. W. Thomas [1] using the TAY code. Thomas essentially reversed the path in parameter space taken by Adair. The four-cell Taylor vortex flow was used as the starting flow at $R_i = 300$ and $R_o = -320$. In the initial experiment, R_o was slowly increased while R_i was kept fixed at 300. The four-cell solution was observed to persist until $R_o = -40$. However, it is believed now that insufficient settling time was allowed and that, in fact, the four-cell solution loses stability at a value of R_o much less than -40. A subsequent experiment was performed to test this, again starting at $R_o = -320$ with the four-cell solution but now using hundreds of thousands of time steps to ensure sufficient settling time. The outer Reynolds number was slowly increased and the four-cell solution was observed to persist until around $R_o = -130$. Because so many time steps were used in this experiment, Thomas is fairly confident that the four-cell vortex flow is stable for $-310 \leq R_o \leq -130$. However, by the time R_o was further increased to -120, and the system was allowed to settle, the flow had drastically changed character and become a disorganized time-dependent flow. However, this time-dependent flow appears to be a different flow than that observed at the same parameter values in the initial run where R_o was decreased. Although

From (R_i, R_o)	*To* (R_i, R_o)	*Observed Flow*
$(300, -320)$	$(300, -130)$	four-cell Taylor vortex flow
$(300, -120)$	$(300, -90)$	time-dependent flow
$(300, -85)$	$(300, 0)$	three-cell Taylor vortex flow
$(295, 0)$	$(295, 0)$	two-cell Taylor vortex flow

Table 4.2: *Summary of axisymmetric flows observed by J. W. Thomas [1].*

further computations need to be done, initially it appears that this disorganized flow is also time-periodic.

As R_o was further increased in this experiment, these time-dependent flows were found to persist to $R_o = -85$, at which point a three-cell Taylor vortex flow was found. This three-cell flow is not time-dependent and is still axisymmetric; however, it does not possess the midplane symmetry of the four- or two-cell flows. A representative plot of this three-cell flow can be found in figure 4.3. The value of R_o was further increased and this three-cell solution persisted until $R_o = 0$. (Recall that Adair observed a two-cell solution at this point in parameter space.) Thomas then decreased the inner Reynolds number very slowly and observed that by $R_i = 295$, the three-cell flow had given way to a two-cell Taylor vortex flow. The transition from this three-cell flow to two-cell flow is rather abrupt.

The results of Thomas' experiments are summarized in table 4.2. It is worth reiterating that all of these observed flows are axisymmetric, a fact verified by using the full three-dimensional capabilities of the TAY code.

4.2.2 Discussion of Bifurcation Aspects

Aside from the variety of axisymmetric flows seen in [2] and the recent experiments of J. W. Thomas in [1], of particular interest are the solution bifurcations that have been observed. As described above, there appears to be a bifurcation that occurs near $R_i = 300$, $R_o = -90$. Presumably, the two-cell Taylor vortex flow loses stability to a time-periodic solution as R_o is decreased to -90 (a Hopf bifurcation).

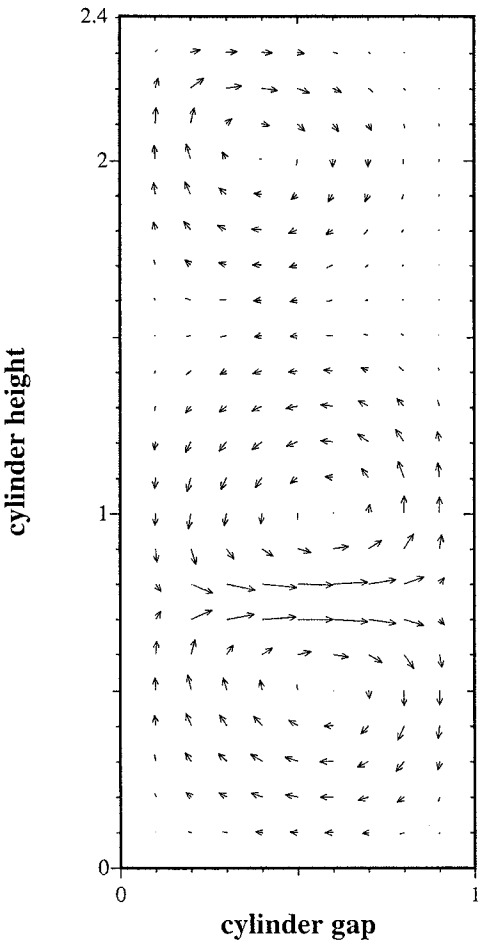

Figure 4.3: *Radial-axial velocity field of a three-cell Taylor vortex flow projected onto a vertical cylinder slice.*

Similarly, this disorganized time-periodic flow appears to lose stability and a four-cell Taylor vortex flow becomes the stable solution as R_o is further decreased to -320. It is obvious that there are also bifurcations occurring in Thomas' experiments, with the four-cell solution presumably losing stability to a time-dependent solution and to an asymmetric (with respect to the axial variable) three-cell solution. This small region of parameter space, the R_oR_i-plane, is ripe with interesting hysteretic behavior.

4.3 Concluding Remarks

In this chapter, we have looked at the Taylor-Couette problem and the time-stepper code TAY written by Adair to calculate fluid flow solutions for this system. We have detailed the numerical algorithms used in the code, and described some of the more interesting axisymmetric flows produced by the program. This code has shown that the small strip of parameter space, $R_i = 300, -320 \leq R_o \leq 0$, is replete with many different fluid flow solutions and solution bifurcations.

Even though the TAY code does a satisfactory job at computing fluid flow solutions, it is less than ideal for performing a detailed bifurcation study of the Taylor-Couette problem. As we demonstrated in Chapter 2, the TAY code can only compute stable solutions of the Navier-Stokes equations, assuming sufficient settling time is allowed. This last caveat is worth noting, particularly in the case of equilibrium solutions. We have run some experiments which require millions of time steps to settle to a steady solution. As we saw, in one experiment Thomas did not use a sufficient number of time steps to accurately obtain an equilibrium solution, which led to misleading results. Because of this, there can be doubt as to whether a computed solution truly is the desired steady solution or whether the system has not yet had sufficient time to equilibrate.

Furthermore, since the TAY code is a time evolution scheme, it is difficult to precisely identify the location in parameter space where bifurcation points occur. The presence of a bifurcation may only be detected by observing qualitative changes in

the solution behavior. These changes are typically visible only farther away from the bifurcation point, because the convergence rate of the time-stepper near a bifurcation point is arbitrarily slow (as shown in Section 2.1.3). In some experiments, for example, we found that millions of time steps were needed to accurately calculate a solution near potential bifurcation points.

Using the TAY code, in general it is difficult to determine whether a bifurcation point or a turning point has been passed, because both involve a qualitative change in the observed solution. At a bifurcation point, the stable solution branch loses its stability, and TAY cannot compute along the unstable branch. Instead it will do one of two things, depending on the type of bifurcation. At a supercritical bifurcation, the TAY code will compute solutions along the stable branch emanating from the bifurcation point. However, at a subcritical bifurcation point there is no emanating stable branch. Instead the TAY code will compute solutions along another stable branch not locally connected to the unstable solution branch (global connections may still exist, however). In this case, the transition to this new branch is typically abrupt.

Now consider the behavior of the code at a turning point. As a turning point is passed, the TAY code "falls off" the stable branch onto some other stable solution branch. This transition, which is also quite sudden, is very similar to that at a subcritical bifurcation point. This makes distinguishing between a subcritical bifurcation point and a turning point difficult. On the other hand, if the bifurcation point is supercritical, additional information may be obtained from TAY (e.g., velocity field or contour plots) that allows the user to conclude that a bifurcation point has been passed rather than a turning point.

Either the outer or the inner Reynolds number plays the role of the bifurcation parameter in all numerical experiments conducted with TAY, since the radius ratio η and the aspect ratio Γ are fixed during experiments. Branches of solutions are parameterized by a Reynolds number. This renders the code incapable of following a solution branch around any turning points. It has no built-in mechanism to deal

with such behavior, such as incorporating an arclength continuation method like that of H. Keller [73].

Finally, the TAY code does not compute the period of any time-periodic flows as part of its solution scheme. An estimate of the period may be obtained only *a posteriori* by plotting the velocity or pressure time series at a specific point in the spatial domain.

Our goal is to use the Newton-Picard method to modify and expand this time-stepper code to make it more suitable for use as a bifurcation tool. Specifically, we wish to use the modified code to perform the following:

- to more accurately pinpoint the values of the Reynolds numbers at which the above observed bifurcations occur;
- to obtain some information as to what type of bifurcations they are; in particular, we wish to confirm what we suspect are Hopf bifurcations;
- to compute unstable solutions;
- to compute the period of time-periodic solutions; and
- to detect turning points and follow solution branches around them.

We now turn to the details of implementing the Newton-Picard method as a computational wrapper around the TAY code.

Chapter 5

Computational Details

In this chapter, we discuss various aspects of implementing the time-stepper based numerical bifurcation code for the Taylor-Couette problem. In particular, we will focus on the changes to the Newton-Picard method that were necessary to form the wrap-around program for the TAY code which we named NPTAY. At the end of the chapter, we will also discuss the tests that were run to verify its accuracy.

5.1 Implementation

5.1.1 Mathematical Context of the Time-Stepper

We begin by considering how to cast Adair's time-stepper code TAY in a form usable by the Newton-Picard method. For this dissertation, we consider only axisymmetric flows in the Taylor-Couette problem, that is, flows that are independent of θ. As a result, the velocity components and pressure are functions of r, z, and t only. We arrange the solution vector so that it is of the form $(\mathbf{u}, \mathbf{v}, \mathbf{w})$; here, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are vectors consisting of all of the radial, azimuthal, and angular velocities, respectively, evaluated at each interior gridpoint of the rz-plane. (The pressure is not included for more technical reasons—see below and Appendix A for details.) Let R denote the total number of radial gridpoints (including boundaries) and let Z denote the total number of axial gridpoints; then the size of each of $\mathbf{u}, \mathbf{v}$, and $\mathbf{w}$ is $(R-2)(Z-2)$, which makes the entire system have dimension $N = 3(R-2)(Z-2)$.

Before continuing, we address two points that arise from the forgoing discussion. First, velocities are evaluated only at the interior gridpoints since the velocities on the boundaries (top, bottom, inside cylinder, and outside cylinder) are entirely determined by the Reynolds numbers R_i and R_o. More specifically, the TAY code uses no-slip boundary conditions everywhere, so all radial and axial boundary velocities are zero, and azimuthal boundary velocities are only nonzero at rotating cylinder walls.

Second, pressure values are not stored as part of the solution vector because, as we mentioned in Chapter 4, the Chorin-Témam scheme can be viewed as a function that maps the current velocity $\mathbf{v}^n$ to the velocity $\mathbf{v}^{n+1}$ at the next time step. Since the pressure is a function of $\mathbf{v}^n$ only, it is used only as an intermediate term in the calculation of $\mathbf{v}^{n+1}$, and therefore does not need to be stored.

Let $\mathbf{x} = (\mathbf{u}, \mathbf{v}, \mathbf{w}) \in \mathbb{R}^N$. The Chorin projection scheme, as part of its solution algorithm, enforces the condition that $\mathbf{x}$ have zero (discrete) divergence. In this light, we can view the Navier-Stokes equations as being in the form

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \tag{5.1}$$

precisely the form to which the Newton-Picard method applies. In terms of time-stepping, we can think of Adair's numerical time integration code as an iteration of the form

$$\mathbf{x}^{n+1} = \mathbf{F}(\mathbf{x}^n), \quad n = 0, 1, \ldots, \tag{5.2}$$

where $\mathbf{F}(\mathbf{x}) = \mathbf{x} + \Delta t \mathbf{f}(\mathbf{x})$. Hence, we are viewing the map $\mathbf{F}$ as the discrete solution operator over one time step, integrating the Navier-Stokes equations forward in time by Δt. (As usual, this notation suppresses the dependence of the solution on the various parameters in the Taylor-Couette problem, specifically R_i and R_o.) The domain of $\mathbf{F}$ is the subspace of $\mathbb{R}^N$ consisting of those vectors with zero (discrete) divergence. As detailed in Chapter 2, equilibrium solutions of (5.1) are fixed points

of (5.2), the convergence of which is determined by the eigenvalues of the Jacobian matrix of $\mathbf{F}$, as given in Proposition 2.4.

As we saw in Section 3.3.3, the computations for calculating time-periodic solutions involve evaluating the term $\phi(\mathbf{u}_0, T)$. Recall that this is simply the solution to the system $d\mathbf{u}/dt = \mathbf{f}(\mathbf{u})$ at time $t = T$ with initial condition $\mathbf{u}(0) = \mathbf{u}_0$. To evaluate $\phi(\mathbf{u}_0, T)$ via the time-stepper (5.2), we simply set $\mathbf{x}^0 = \mathbf{u}_0$ and iterate a total of $k = T/\Delta t$ times. Time-periodic solutions of (5.1) are therefore fixed points of the map $\mathbf{G} = \mathbf{F}^k$, whose stability is determined by the eigenvalues of the Monodromy matrix (see Theorem 2.5).

Having cast Adair's time-stepper code in this way, we immediately see how the Newton-Picard method can be applied. For equilibrium solutions, we use the techniques detailed in Section 3.3.1, while for time-periodic solutions, we use those found in Section 3.3.3.

5.1.2 Details of the Implementation

We now detail how the wrap-around code NPTAY was implemented and show how its implementation differs from the discussion of Chapter 3. Two themes permeate our discussion in this section. First, there are slight changes to the original Newton-Picard method incorporated in the final version of the code. Second, the original paradigm of viewing the TAY code as a true black-box had to be abandoned as we gained more insight into the problem and the specifics of the solution scheme.

Since the TAY code was written in C++, we wrote NPTAY in C++ for maximum compatibility. Data structures were set up as a communication interface between NPTAY and TAY. This involved opening the TAY black-box and implementing this interface directly within the code, rather than using text files to send information back and forth. This was done for efficiency reasons.

Changes to the Newton-Picard Method

One of the biggest departures the code takes from the original Newton-Picard method is with regard to computing an orthonormal basis for the invariant subspace $\mathcal{P}$. As Lust suggests, this basis is formed by constructing Schur vectors which span $\mathcal{P}$, rather than (generalized) eigenvectors [85]. However, we do not use subspace iteration as Lust does. Instead, the software package ARPACK [82] is used to compute the dominant eigenvalues of the Jacobian matrix and the corresponding Schur vectors. ARPACK is written in FORTRAN, so this required some additional programming to successfully have our C++ program communicate with these FORTRAN routines. The choice to use ARPACK was made based on poor convergence results we obtained in our initial implementation of the subspace iteration algorithm.

The algorithms used in ARPACK are based on Krylov subspace methods, specifically, the implicitly restarted Arnoldi method of Sorensen [124]. (The package's name comes from ARnoldi PACKage.) In fact, this implicitly restarted Arnoldi method is related to subspace iteration, but its convergence properties for large-scale eigenvalue problems are superior [81].

Another difference between the implementation and the original Newton-Picard concept involves the number of dominant eigenvalues constructed. Lust suggests setting a small tolerance $\delta > 0$ and computing all eigenvalues whose complex modulus is greater than $1-\delta$. This technique works well if the spectrum of the Jacobian is well separated. However, preliminary results showed that the spectrum of every Jacobian we computed was not well separated and, more troublesome, that there were many eigenvalues clustered around 1. Indeed, it was not unusual to find that over half of the eigenvalues had modulus greater than 0.99. Thus, rather than computing all eigenvalues greater than a certain value $1-\delta$ (which in our situation would necessitate a very small value of δ) we felt it was easier to use a fixed basis size for the invariant subspace $\mathcal{P}$. Typically, we used a basis size of 15, which we felt was sufficiently large

to capture all dangerous eigenvalues, but small enough that computations in $\mathcal{P}$ were efficient.

As discussed in Section 3.3.3, to compute a periodic solution of (5.1), some sort of phase condition is necessary. Lust states that instead of an explicit phase condition, one can be imposed implicitly by solving the small system which determines $\Delta\bar{\mathbf{p}}$ by a least squares method. Thus, rather than solving

$$\begin{bmatrix} V_p^T(M^\nu - I)V_p & V_p^T\phi_T^\nu \\ \psi_{\mathbf{u}}^\nu V_p & \psi_T^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T[\mathbf{r}(\mathbf{u}_0^\nu, T^\nu) + M^\nu V_q \Delta\bar{\mathbf{q}}^\nu] \\ \psi(\mathbf{u}_0^\nu, T^\nu) + \psi_{\mathbf{u}}^\nu V_q \Delta\bar{\mathbf{q}}^\nu \end{bmatrix},$$

which involves the phase condition and its derivatives, we instead solve the underdetermined system

$$\begin{bmatrix} V_p^T(M^\nu - I)V_p & V_p^T\phi_T^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T[\mathbf{r}(\mathbf{u}_0^\nu, T^\nu) + M^\nu V_q \Delta\bar{\mathbf{q}}^\nu] \end{bmatrix} \tag{5.3}$$

by a least squares method. This is done by utilizing the LAPACK [7] routine DGELSS, which solves the least squares problem using the SVD of the matrix. The least squares solution returned is the one with minimal norm, and it is this imposition of minimal norm that implicitly determines a phase condition. See Lust [85] for more details on how this condition is close to an optimal one.

We also made a change in how the derivative ϕ_T is computed. For equilibrium solutions, this quantity is obviously zero, but its evaluation is necessary when computing time-periodic solutions. Lust suggests using the fact that $\phi(\mathbf{u}(0), T)$ is a solution of (5.1) at time T with initial condition $\mathbf{u}_0$ to conclude that

$$\left.\frac{d\phi}{dT}\right|_{(\mathbf{u}_0, T)} = \mathbf{f}(\phi(\mathbf{u}_0, T)).$$

This works well when the right-hand side $\mathbf{f}$ of the dynamical system (5.1) is easily available. This is not true in our case, since an explicit formula for the right-hand side is difficult to extract from the TAY code. We found it simpler and effective to use a finite difference approximation for this derivative:

$$\left.\frac{d\phi}{dT}\right|_{(\mathbf{u}_0, T)} \approx \frac{\phi(\mathbf{u}_0, T+\varepsilon) - \phi(\mathbf{u}_0, T)}{\varepsilon}. \tag{5.4}$$

Typically, we let $\varepsilon = \Delta t$, the time step increment used in TAY.

Finally, consider the Picard iteration (3.15), which involves solving the $\mathcal{Q}$-equations for $\Delta\mathbf{q}$. The user-defined parameter ℓ determines how many times this iteration is performed. Lust, in the small-dimensional problems he studied in [85], states that $\ell = 1$ is sufficient for convergence. We found that not to be true in our case. In fact, we found that at least one thousand Picard iterations were needed typically.

We believe that such a large number of Picard iterations are necessary because Adair's scheme is very weakly contractive. Recall that the eigenvalue analysis above showed that typically more than half of all eigenvalues have modulus greater than 0.99. The error in using the kth partial sum of the Neumann series to approximate $(V_q^T M V_q - I)^{-1}$ is bounded by

$$|\text{error}| \leq \frac{\|V_q^T M V_q\|^k}{1-\rho} \|\mathbf{r}\|,$$

where ρ denotes the spectral radius of $V_q^T M V_q$. Typical values of these variables are $\rho \approx 0.999$ and $\|\mathbf{r}\| \approx 10^{-6}$. Using the approximation $\|V_q^T M V_q\| \approx \rho$ (in fact, we only know $\rho \leq \|V_q^T M V_q\|$), we see that using $k = 1000$ Picard iterations gives an error bound of ≈ 0.0004.

As the Newton iterations near convergence, the value of $\|\mathbf{r}\|$ decreases, and hence fewer Picard iterations are required to obtain the solution of the $\mathcal{Q}$-equations to the same accuracy. To take advantage of this fact, the NPTAY code performs Picard iterations until the norm of the difference of successive iterates falls below a user-defined tolerance, rather than iterating a fixed number of times. This prevents the code from performing unnecessary iterations.

In our tests, we found that a very small Picard convergence tolerance is needed for the Newton's method algorithm to converge. Typically, the Picard convergence tolerance needed to be two or three orders of magnitude smaller than the Newton convergence tolerance. In other words, the $\mathcal{Q}$-equations

$$(V_q^T (M^\nu - I) V_q) \Delta\bar{\mathbf{q}}^\nu = -V_q^T \mathbf{r}(\mathbf{u}^\nu)$$

must be solved very accurately to get an accurate answer for $\Delta\mathbf{u}$. Since the value of $\Delta\tilde{\mathbf{q}}$ is used in the solution of the $\mathcal{P}$-equations (5.3), this suggests that the system defining the $\mathcal{Q}$-equations is poorly conditioned. Indeed, we can easily see that it is. Since the columns of V_q are orthonormal, the matrix $V_q^T(M^\nu - I)V_q$ is ill-conditioned if and only if $M^\nu - I$ is ill-conditioned. Because there are so many eigenvalues of the Jacobian matrix M clustered about 1 in the complex plane, we see that the matrix $M^\nu - I$ is nearly singular and hence is very ill-conditioned.

Jacobian-Vector Products

To find a basis for the invariant subspace $\mathcal{P}$, the ARPACK software requires that we be able to calculate Jacobian-vector products. The same is true for every Picard iteration subsequent to the first. More precisely, we need only to be able to compute the *multiplicative action* of the Jacobian matrix M given an arbitrary vector $\mathbf{v}$; we need not have the matrix M itself at our disposal (which, in fact, we do not). One of the nice features of ARPACK is that it only requires the matrix action, not the matrix itself, to compute dominant eigenvalues and corresponding Schur vectors. The method used in NPTAY to compute this matrix action depends on the type of solution being sought—equilibrium or time-periodic.

Since the Jacobian matrix is not available explicitly, a straightforward way to compute Jacobian-vector products is via finite differences based on directional derivatives. This has the advantage of only requiring additional calls to the time-stepper. Let M denote the Jacobian matrix evaluated at a solution vector $\mathbf{u}$ (and at time T for the time-periodic case). For equilibrium solutions, a first order approximation of the matrix-vector product $M\mathbf{v}$ is

$$M\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{u}+\varepsilon\mathbf{v}) - \mathbf{F}(\mathbf{u})}{\varepsilon}. \tag{5.5}$$

Similarly, a first order approximation for time-periodic solutions is

$$M\mathbf{v} \approx \frac{\boldsymbol{\phi}(\mathbf{u}+\varepsilon\mathbf{v}, T) - \boldsymbol{\phi}(\mathbf{u}, T)}{\varepsilon}, \tag{5.6}$$

where $\phi(\mathbf{u}, T)$ is the solution of $d\mathbf{u}/dt = \mathbf{f}(\mathbf{u})$ at time $t = T$ for the initial condition $\mathbf{u}$. Using either of these finite difference formulas, each matrix-vector product requires a call to the TAY code, in addition to the one-time call needed to compute $\mathbf{F}(\mathbf{u})$ or $\phi(\mathbf{u}, T)$. As discussed above, a large number of Picard iterations typically are required, so the code requires an equally large number of calls to TAY. Unfortunately, this causes the code to be slow. Although we are able to compute accurate solutions with this method, the typical run time for one solution computation is too long for a practical bifurcation code. Changes therefore were made to the implementation of Jacobian-vector products in NPTAY to speed up the program.

Initial tests using (5.5) for the computation of equilibrium solutions failed to accurately compute the dominant eigenvalues of the Jacobian matrix. Instead, several spurious eigenvalues of large modulus were calculated along with the correct dominant eigenvalues. Only by using knowledge of the solution algorithm buried within the TAY code were we later able to identify the cause of these spurious eigenvalues. Without this knowledge of the inner workings of the time-stepper, we would not have been able to resolve this issue. Appendix A discusses these spurious eigenvalues and their cause in more detail.

In the meantime, we sought an alternative to using finite differences for computing Jacobian-vector products. Our solution was to compute the action of the Jacobian matrix in a more direct manner: by explicitly linearizing the Navier-Stokes equations and solving them by applying the same solution technique and discretization scheme as used in the TAY code. This resulted in a linearized version of TAY. Jacobian-vector products involve calls to this linearized TAY code rather than calls to the original code. Writing this linearized code involved opening the black-box of the time-stepper, which we felt was unavoidable in this case. Since much effort has been spent writing this linearized code, it has been left in the final version of the NPTAY code, even though the cause of the spurious eigenvalues has been identified. Details of the linearized solver, including its discretization, can be found in Appendix B.

Although it did solve the initial eigenvalue difficulties we had, the linearized code did not rectify the slow computation speed of NPTAY. Hence, the other alteration made to TAY was to completely break open the black-box and rewrite part of its solver routine. Using computer timing routines, we found that the vast majority of the time spent within TAY (and in the linearized solver) was the Gauss-Seidel algorithm used to compute the pressures from the Poisson equation (4.5). This equation must be solved regardless of whether we are using finite differences or the linearized solver to compute Jacobian-vector products. Upon discretization of the pressure and velocities, solving this equation amounts to solving a matrix equation. For a fixed cylinder geometry, the matrix of coefficients of this matrix equation is always the same, regardless of the inner or outer Reynolds numbers. Rather than using the existing Gauss-Seidel solver in TAY, we instead utilize a matrix inverse to solve the discrete Poisson equation for the pressure. There are some technical details of this procedure whose complete exposition the reader will find in Appendix C. The upshot is that, rather than using an iterative method to solve the Poisson equation for the pressure (which in practice can take several hundred thousand iterations per solve), we reduce the solution to a simple matrix-vector multiplication, which can be done quickly using existing fast matrix-vector routines. We chose to use the standard BLAS matrix-vector multiply routine DGEMV [45, 46] to implement this.

The situation differs slightly for time-periodic computations. Each Jacobian-vector product requires the evaluation of $\phi(\mathbf{u}+\varepsilon\mathbf{v}, T)$; in particular, it requires integrating the system over the time interval $[0, T]$. If $\Delta t = 5 \times 10^{-5}$ and $T = 0.1$, for instance, this would require 2000 matrix-vector products to compute one Jacobian-vector product. (These matrix-vector products are the ones used in the redesigned Poisson equation solver.) However, the eigenvalues of the Jacobian matrix $\phi_{\mathbf{u}}$ are typically well-separated, and hence fewer Picard iterations are needed during each Newton iteration. Applying finite difference equation (5.6) to approximate Jacobian-vector products therefore is quite reasonable, and is used in the final version of the NPTAY code.

Finally, to speed up the code even more, the Jacobian matrix is held fixed at the beginning of the Newton iterations. Specifically, the basis for the small-dimensional subspace $\mathcal{P}$ is held fixed. This "simplified Newton's method" sacrifices the typical quadratic convergence of Newton's method for linear convergence. The basis for $\mathcal{P}$ is recomputed only if the norm of $\Delta\mathbf{u}$ increases from one Newton iteration to the next. We found that the time saved by this method more than made up for the increased number of Newton iterations needed for convergence.

Arclength Continuation

For computations that require traversing a turning point, Keller's arclength continuation strategy [73] is incorporated into the NPTAY code, as described in Section 3.4. Let us first consider the problem of computing a branch of equilibrium solutions.

Let s denote the new parameter, which represents an approximation to arclength along the solution branch as measured from some starting solution. There are many parameterization equations that can be used to provide this approximation to arclength. Our particular equation requires that we know the previous two solutions on the branch: $(\mathbf{u}_{-1}, \lambda_{-1}) = (\mathbf{u}(s_{-1}), \lambda(s_{-1}))$ and $(\mathbf{u}_0, \lambda_0) = (\mathbf{u}(s_0), \lambda(s_0))$, where $s_{-1} < s_0$. We form the secant vector $(\boldsymbol{\tau}, \sigma)$, where

$$\boldsymbol{\tau} = \frac{\mathbf{u}_0 - \mathbf{u}_{-1}}{s_0 - s_{-1}}$$

and

$$\sigma = \frac{\lambda_0 - \lambda_{-1}}{s_0 - s_{-1}}.$$

These are approximations to the derivatives $d\mathbf{u}/ds$ and $d\lambda/ds$ evaluated at $s = s_0$. Let $s = s_0 + \Delta s$ for some Δs. The parameterization equation used for computing the solution at $s_0 + \Delta s$ is

$$n(\mathbf{u}(s), \lambda(s), s) = \theta\boldsymbol{\tau}^T(\mathbf{u}(s) - \mathbf{u}(s_0)) + (1-\theta)\sigma(\lambda(s) - \lambda(s_0)) - (s - s_0).$$

The user-defined parameter θ plays the role of a weighting constant. Typically, $\theta = 0.9$ worked well for our experiments.

The parameterization equation for time-periodic solutions is very similar. The secant vector is of the form (τ, π, σ), where τ and σ are defined as above and

$$\pi = \frac{T(s_0) - T(s_{-1})}{s_0 - s_{-1}}.$$

The parameterization equation is

$$n(\mathbf{u}_0(s), T(s), \lambda(s), s) = \theta_{\mathbf{u}} \tau^T (\mathbf{u}_0(s) - \mathbf{u}_0(s_0)) + \theta_T \pi (T(s) - T(s_0)) + \theta_\lambda \sigma (\lambda(s) - \lambda(s_0)) - (s - s_0),$$

where $\theta_{\mathbf{u}} + \theta_T + \theta_\lambda = 1$.

As with other derivatives, the derivatives $\mathbf{F}_\lambda$ and ϕ_λ are also computed by finite differences. For $\mathbf{F}_\lambda^\nu$, we use

$$\mathbf{F}_\lambda^\nu \approx \frac{\mathbf{F}(\mathbf{u}^\nu, \lambda^\nu + \epsilon_\lambda) - \mathbf{F}(\mathbf{u}^\nu, \lambda^\nu)}{\epsilon_\lambda},$$

while ϕ_λ^ν is approximated by the finite difference

$$\phi_\lambda^\nu \approx \frac{\phi(\mathbf{u}^\nu, T^\nu, \lambda^\nu + \epsilon_\lambda) - \phi(\mathbf{u}^\nu, T^\nu, \lambda^\nu)}{\epsilon_\lambda}.$$

Typically we used $\epsilon_\lambda = 0.1$.

The NPTAY code also incorporates a variable-length step size. If the Newton-Picard method fails to converge at $s_0 + \Delta s$, then the size of Δs is halved, and an attempt is made to compute the solution at $s_0 + \Delta s/2$. The step size can be repeatedly halved as necessary until a minimum (user-defined) step size is reached. Decreasing the step size typically is needed as a turning point is approached.

5.1.3 The Final Program

In this section, we summarize the NPTAY code as it appears in its final form. It can be thought of two separate programs, one to compute equilibrium solutions and one to compute time-periodic solutions, each of which may be run in parameter

continuation or arclength continuation mode. At the core of NPTAY lies the modified version of the time-stepper TAY. In addition to minor modifications to improve communication and data transfer between NPTAY and TAY, the Gauss-Seidel solver for the Poisson equation (4.5) in TAY has been replaced with multiplication by an appropriate matrix inverse (see Appendix C for details). Further modifications to TAY include the addition of several routines that solve the linearized Navier-Stokes equations to evaluate the multiplicative action of the Jacobian matrix directly rather than using a finite difference approximation.

After initial tests, we found the following arrangements result in the fastest code:

- For computing equilibrium solutions, the multiplicative action of the Jacobian is evaluated via the linearized Navier-Stokes solver we added to the TAY code. All Jacobian-vector products, both for computing the basis of the invariant subspace $\mathcal{P}$ and for Picard iterations, are computed this way. The Jacobian matrix is never constructed explicitly.

- For time-periodic solutions, Jacobian-vector products are performed by using finite difference equation (5.6). As with equilibrium solutions, the Jacobian matrix is never constructed explicitly.

For either type of problem, the basic procedure is to first compute the flow solution at particular Reynolds numbers (R_i, R_o) using the original TAY code. This solution acts as the initial solution guess when one of the Reynolds numbers is changed. The NPTAY code, based on this initial guess, computes a basis for the invariant subspace $\mathcal{P}$ and proceeds to compute the solution at the new (R_i, R_o) values using the Newton-Picard method. This process is then repeated for new values of (R_i, R_o). This procedure works well far from turning points. As a turning point is approached, this method will break down and, typically, the Newton-Picard method will fail to converge. In this case, the NPTAY code can be run in arclength continuation mode to traverse the turning point.

Below is pseudo-code for the NPTAY algorithm used to compute equilibrium solutions.

Algorithm *NPTAY for Equilibrium Solutions*
Input:
Starting solution $\mathbf{u}^0$
Parameter values (R_i, R_o)
Basis size p
Maximum number of Newton iterations MAXITER
Picard iteration convergence tolerance PTOL
Maximum number of Picard iterations per Newton step ℓ
Convergence tolerance TOL
Output: Solution $\mathbf{u}$ at (R_i, R_o)

Use ARPACK to compute an orthonormal basis V_p for the p-dimensional dominant invariant subspace of the Jacobian matrix $M = M(\mathbf{u}^0)$
$\nu \leftarrow 0$
CVG $\leftarrow$**false**
do
    Compute $\mathbf{F}(\mathbf{u}^\nu)$
    $\mathbf{r}^\nu \leftarrow \mathbf{F}(\mathbf{u}^\nu) - \mathbf{u}^\nu$
    Solve the $\mathcal{Q}$-equations for $\Delta\mathbf{q}^\nu$ via Picard iteration:
        $\Delta\mathbf{q}_{\text{old}} \leftarrow (I - V_p V_p^T)\mathbf{r}^\nu$
        $k \leftarrow 2$
        while $(k < \ell)$ **and** (norm $>$ PTOL)
            $\Delta\mathbf{q}_{\text{new}} \leftarrow (I - V_p V_p^T)(M\Delta\mathbf{q}_{\text{old}} + \mathbf{r}^\nu)$
            norm $\leftarrow \|\Delta\mathbf{q}_{\text{new}} - \Delta\mathbf{q}_{\text{old}}\|$
            $\Delta\mathbf{q}_{\text{old}} \leftarrow \Delta\mathbf{q}_{\text{new}}$
            $k \leftarrow k + 1$
        end while
        $\Delta\mathbf{q}^\nu \leftarrow \Delta\mathbf{q}_{\text{new}}$
    Solve the $\mathcal{P}$-equations (3.16) for $\Delta\bar{\mathbf{p}}^\nu$
    $\Delta\mathbf{u}^\nu \leftarrow V_p\Delta\bar{\mathbf{p}}^\nu + \Delta\mathbf{q}^\nu$
    $\mathbf{u}^{\nu+1} \leftarrow \mathbf{u}^\nu + \Delta\mathbf{u}^\nu$
    Compute $\mathbf{F}(\mathbf{u}^{\nu+1})$
    $\mathbf{r}^{\nu+1} \leftarrow \mathbf{F}(\mathbf{u}^{\nu+1}) - \mathbf{u}^{\nu+1}$
    if $(\|\Delta\mathbf{u}^\nu\| < \text{TOL})$ **and** $(\|\mathbf{r}^{\nu+1}\| < \text{TOL})$
      then CVG $\leftarrow$**true**
    if $\|\Delta\mathbf{u}^\nu\| > \|\Delta\mathbf{u}^{\nu-1}\|$
      then recompute basis V_p for the Jacobian matrix $M(\mathbf{u}^{\nu+1})$
    $\nu \leftarrow \nu + 1$
while $(\nu <$ MAXITER$)$ **and** (CVG = **false**)
if (CVG = **false**)
  then return error
  else return $\mathbf{u} = \mathbf{u}^\nu$

Next, we present pseudo-code for the NPTAY program for computing time-periodic solutions.

Algorithm *NPTAY for Time-Periodic Solutions*
Input:
Starting solution $\mathbf{u}^0$, starting period T^0
Parameter values (R_i, R_o)
Basis size p
Maximum number of Newton iterations MAXITER
Picard iteration convergence tolerance PTOL
Maximum number of Picard iterations per Newton step ℓ
Convergence tolerance TOL
Output: Solution $\mathbf{u}_0$ at time 0 and period T at (R_i, R_o)

1. Use ARPACK to compute an orthonormal basis V_p for the p-dimensional dominant invariant subspace of $M = M(\mathbf{u}^0, T^0)$
2. $\nu \leftarrow 0$
3. CVG $\leftarrow$**false**
4. **do**
5. Compute $\phi(\mathbf{u}^\nu, T^\nu)$
6. $\mathbf{r}^\nu \leftarrow \phi(\mathbf{u}^\nu, T^\nu) - \mathbf{u}^\nu$
7. Solve the $\mathcal{Q}$-equations for $\Delta\mathbf{q}^\nu$ via Picard iteration:
8. $\Delta\mathbf{q}_{\text{old}} \leftarrow (I - V_p V_p^T)\mathbf{r}^\nu$
9. $k \leftarrow 2$
10. **while** $(k < \ell)$ **and** (norm $>$ PTOL)
11. $\Delta\mathbf{q}_{\text{new}} \leftarrow (I - V_p V_p^T)(M\Delta\mathbf{q}_{\text{old}} + \mathbf{r}^\nu)$
12. norm $\leftarrow \|\Delta\mathbf{q}_{\text{new}} - \Delta\mathbf{q}_{\text{old}}\|$
13. $\Delta\mathbf{q}_{\text{old}} \leftarrow \Delta\mathbf{q}_{\text{new}}$
14. $k \leftarrow k + 1$
15. **end while**
16. $\Delta\mathbf{q}^\nu \leftarrow \Delta\mathbf{q}_{\text{new}}$.
17. Approximate the derivative ϕ_T^ν by finite difference (5.4)
18. Solve the $\mathcal{P}$-equations

$$\begin{bmatrix} V_p^T(M - I)V_p & V_p^T\phi_T^\nu \end{bmatrix} \begin{bmatrix} \Delta\bar{\mathbf{p}}^\nu \\ \Delta T^\nu \end{bmatrix} = - \begin{bmatrix} V_p^T[\mathbf{r}^\nu + M\Delta\mathbf{q}^\nu] \end{bmatrix}$$

for $\Delta\bar{\mathbf{p}}^\nu$ and ΔT^ν using a least squares method based on the SVD.

19. $\Delta\mathbf{u}^\nu \leftarrow V_p\Delta\bar{\mathbf{p}}^\nu + \Delta\mathbf{q}^\nu$
20. $\mathbf{u}^{\nu+1} \leftarrow \mathbf{u}^\nu + \Delta\mathbf{u}^\nu$
21. $T^{\nu+1} \leftarrow T^\nu + \Delta T^\nu$
22. Compute $\phi(\mathbf{u}^{\nu+1}, T^{\nu+1})$
23. $\mathbf{r}^{\nu+1} \leftarrow \phi(\mathbf{u}^{\nu+1}, T^{\nu+1}) - \mathbf{u}^{\nu+1}$
24. **if** $(\|\Delta\mathbf{u}^\nu\| < \text{TOL})$ **and** $(\|\mathbf{r}^{\nu+1}\| < \text{TOL})$
25. **then** CVG $\leftarrow$**true**
26. **if** $\|\Delta\mathbf{u}^\nu\| > \|\Delta\mathbf{u}^{\nu-1}\|$

27. **then** reconstruct the Jacobian matrix $M = M(\mathbf{u}^{\nu+1}, T^{\nu+1})$
28. recompute basis V_p for M
29. $\nu \leftarrow \nu + 1$
30. **while** ($\nu <$ MAXITER) **and** (CVG = **false**)
31. **if** (CVG = **false**)
32. **then return** error
33. **else return** $\mathbf{u} = \mathbf{u}^{\nu}$ and $T = T^{\nu}$

Finally, we present the NPTAY algorithm as used with arclength continuation for equilibrium solutions. The time-periodic version is similar.

Algorithm *NPTAY for Equilibrium Solutions with Arclength Continuation*
Input:
Starting solutions $(\mathbf{u}(s_{-1}), \lambda(s_{-1}))$ and $(\mathbf{u}(s_0), \lambda(s_0))$, where $\lambda = R_i$ or R_o
Initial step size Δs
Minimum allowed step size $\Delta s_{\min}$
Basis size p
Maximum number of Newton iterations MAXITER
Picard iteration convergence tolerance PTOL
Maximum number of Picard iterations per Newton step ℓ
Convergence tolerance TOL
Output: Solution $\mathbf{u}$ at (R_i, R_o)
1. Use ARPACK to compute an orthonormal basis V_p for the p-dimensional dominant invariant subspace of the Jacobian matrix $M = M(\mathbf{u}(s_0), \lambda(s_0))$
2. Form secant vector $(\boldsymbol{\tau}, \sigma)$
3. $\mathbf{u}^0 \leftarrow \mathbf{u}(s_0)$ and $\lambda^0 \leftarrow \lambda(s_0)$
4. $s \leftarrow s_0 + \Delta s$
5. $\nu \leftarrow 0$
6. CVG $\leftarrow$**false**
7. **do**
8. Compute $\mathbf{F}(\mathbf{u}^{\nu}, \lambda^{\nu})$
9. $\mathbf{r}^{\nu} \leftarrow \mathbf{F}(\mathbf{u}^{\nu}, \lambda^{\nu}) - \mathbf{u}^{\nu}$
10. Compute $\mathbf{F}_{\lambda}^{\nu}$
11. Solve the $\mathcal{Q}$-equations for $\Delta\mathbf{q}_r^{\nu}$ via Picard iteration:
12. $\Delta\mathbf{q}_{\text{old}} \leftarrow (I - V_p V_p^T)\mathbf{r}^{\nu}$
13. $k \leftarrow 2$
14. **while** ($k < \ell$) **and** (norm $>$ PTOL)
15. $\Delta\mathbf{q}_{\text{new}} \leftarrow (I - V_p V_p^T)(M\Delta\mathbf{q}_{\text{old}} + \mathbf{r}^{\nu})$
16. norm $\leftarrow \|\Delta\mathbf{q}_{\text{new}} - \Delta\mathbf{q}_{\text{old}}\|$
17. $\Delta\mathbf{q}_{\text{old}} \leftarrow \Delta\mathbf{q}_{\text{new}}$
18. $k \leftarrow k + 1$
19. **end while**
20. $\Delta\mathbf{q}_r^{\nu} \leftarrow \Delta\mathbf{q}_{\text{new}}$.

```
21.      Solve the Q-equations for Δq_λ^ν via Picard iteration:
22.           Δq_old ← (I − V_p V_p^T) F_λ^ν
23.           k ← 2
24.           while (k < ℓ) and (norm > PTOL)
25.                Δq_new ← (I − V_p V_p^T)(M Δq_old + F_λ^ν)
26.                norm ←‖Δq_new − Δq_old‖
27.                Δq_old ← Δq_new
28.                k ← k + 1
29.           end while
30.           Δq_λ^ν ← Δq_new.
31.      Solve the P-equations (3.28) for Δp̄^ν and Δλ^ν
32.      Δu^ν ← V_p Δp̄^ν + Δq_r^ν + Δq_λ^ν Δλ^ν
33.      u^{ν+1} ← u^ν + Δu^ν and λ^{ν+1} ← λ^ν + Δλ^ν
34.      Compute F(u^{ν+1}, λ^{ν+1})
35.      r^{ν+1} ← F(u^{ν+1}, λ^{ν+1}) − u^{ν+1}
36.      if (‖Δu^ν‖ < TOL) and (‖r^{ν+1}‖ < TOL)
37.        then CVG ←true
38.      if ‖Δu^ν‖ > ‖Δu^{ν−1}‖
39.        then recompute basis V_p for the Jacobian matrix M(u^{ν+1})
40.      ν ← ν + 1
41. while (ν < MAXITER) and (CVG = false)
42. if (CVG = false)
43.   then Δs ← Δs/2
44.        if (Δs < Δs_min)
45.          then return error
46.          else  goto 3
47.   else  return u = u^ν, λ = λ^ν
```

5.2 Verification of the Code

In this section we describe the test problems to which we applied the NPTAY code to test its accuracy. Each test examines a different aspect of the algorithm.

5.2.1 Simple Test Problem for Basic Algorithmic Correctness

The first test problem was used to verify the correctness of the algorithm; in particular, the eigenvalue solver and basis computation, the Newton iteration, and the Picard iteration. Rather than use the complex TAY code as the time-stepper, we chose a simpler test problem for this purpose and formulated an iterative solution algorithm for it. This algorithm played the role of the time-stepper code around which we wrapped the program.

Consider the simple one-dimensional boundary value problem: $u''(x) = 0$ on $(0,1)$, $u(0) = 0$, $u(1) = 3$. Its solution is clearly $u(x) = 3x$. Discretize the problem by dividing the interval $(0,1)$ into N subintervals of equal width $\Delta x = 1/N$ by introducing a partition $\{0 = x_0, x_1, \ldots, x_{N-1}, x_N = 1\}$ of $(0,1)$. Denote $u(x_j)$ by u_j, so that the discrete solution of the problem can be written as the vector $\mathbf{u} = (u_1, u_2, \ldots, u_{N-1})^T$. The second derivative $u''(x)$ is approximated by the typical second-order centered difference equation

$$u''(j\Delta x) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2}.$$

Substituting the boundary conditions $u_0 = 0$ and $u_N = 3$ in the above formula as appropriate, we see that the solution $\mathbf{u}$ satisfies the matrix equation $A\mathbf{u} = \mathbf{f}$, where

$$A = \begin{bmatrix} -2 & 1 & 0 & \cdots & \\ 1 & -2 & 1 & 0 & \cdots \\ & \ddots & \ddots & \ddots & \\ \cdots & 0 & 1 & -2 & 1 \\ & \cdots & 0 & 1 & -2 \end{bmatrix}$$

is an $(N-1) \times (N-1)$ matrix and $\mathbf{f} = (0, \ldots, 0, -3)^T$ is also of size $N-1$.

The iterative algorithm acting as the time-stepper in this test is the successive overrelaxation scheme (SOR scheme) for solving the matrix equation $A\mathbf{u} = \mathbf{f}$. The SOR algorithm is a specific type of a more general category of linear solvers called residual correction schemes. Let us briefly recall the details of such schemes. Suppose $\mathbf{w}$ is an approximation to the solution vector $\mathbf{u}$. How $\mathbf{w}$ is obtained is immaterial for the moment. The residual is defined as the vector $\mathbf{r} = \mathbf{f} - A\mathbf{w}$. Let $\mathbf{e} = \mathbf{u} - \mathbf{w}$ denote the error vector. We then have

$$A\mathbf{e} = A(\mathbf{u} - \mathbf{w}) = A\mathbf{u} - A\mathbf{w} = \mathbf{f} - A\mathbf{w} = \mathbf{r},$$

and hence the exact solution $\mathbf{u}$ is

$$\mathbf{u} = \mathbf{e} + \mathbf{w} = \mathbf{w} + A^{-1}\mathbf{r}. \tag{5.7}$$

Obviously we do not know A^{-1} (otherwise the solution is trivial to compute), so the idea behind residual correction schemes is to approximate A^{-1} by a simpler matrix B. Given an initial guess $\mathbf{w}_0$, the iteration in a general residual correction scheme is to compute

$$\mathbf{w}_{k+1} = \mathbf{w}_k + B\mathbf{r}_k, \quad k = 0, 1, \ldots, \tag{5.8}$$

where $\mathbf{r}_k = \mathbf{f} - A\mathbf{w}_k$. Note that the iteration is simply the analog of (5.7) with A^{-1} replaced by B. This iteration is continued until the norm of successive differences $\|\mathbf{w}_{k+1} - \mathbf{w}_k\|$ is less than some tolerance. Mathematically, in exact arithmetic, we are seeking a fixed point of the iteration (5.8). We can formulate this more precisely by noting that

$$\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k + B\mathbf{r}_k \\
&= \mathbf{w}_k + B(\mathbf{f} - A\mathbf{w}_k) \\
&= (I - BA)\mathbf{w}_k + B\mathbf{f}.
\end{aligned}$$

Hence, we can see that a residual correction scheme is an iterative method of the form (5.8) for computing the fixed point of the map $\mathbf{F}(\mathbf{u}) = (I - BA)\mathbf{u} + B\mathbf{f}$. This is precisely the context to which the Newton-Picard method can be applied.

Write the matrix A as $A = L + D + U$, where L is the lower triangular portion of A below the main diagonal, D is the diagonal matrix comprised of the main diagonal of A, and U is the upper triangular portion of A above the main diagonal. Let ω denote a user-defined parameter such that $0 < \omega < 2$. The SOR scheme is obtained by choosing $B = \omega(D + \omega L)^{-1}$ in (5.8). Incorporating the boundary conditions, this results in the following algorithm, which computes $\mathbf{u}^* = \mathbf{F}(\mathbf{u})$, given $\mathbf{u}$:

Algorithm *SOR Method*
1. $u_1^* = \omega u_2 + (1 - \omega)u_1$
2. **for** $j \leftarrow 2$ **to** $N - 2$
3. $\quad u_j^* = \omega(\frac{1}{2}u_{j+1} + \frac{1}{2}u_{j-1}^*) + (1 - \omega)u_j$
4. **end for**
5. $u_{N-1}^* = \omega(\frac{3}{2} + \frac{1}{2}u_{N-2}^*) + (1 - \omega)u_{N-1}$

Note that the Jacobian matrix is $\mathbf{F}_{\mathbf{u}}(\mathbf{u}) = I - BA$ for any $\mathbf{u}$, so the eigenvalues of the Jacobian can be checked against the eigenvalues of $I - BA = I - \omega(D + \omega L)^{-1}A$. The latter are easy to compute using MATLAB. This provides a way of verifying the accuracy of the eigenvalue solver. For this test, the multiplicative action of the Jacobian matrix was implemented via the finite difference formula (5.5), where the action of $\mathbf{F}$ is given by the SOR algorithm above.

Tests were run for $N = 20$, 50, and 100 gridpoints, using various values of ω. In fact, for some program runs, ω was treated as a varying parameter, playing the same role as the Reynolds number does in the final code. In all cases, the code converged to the correct (discrete) solution $u(x) = 3x$ and all eigenvalue calculations were correct, as compared with the eigenvalues returned by MATLAB. We also observed the following:

- For fixed ω and N, the number of Newton iterations required for convergence decreased as the basis size of the invariant subspace $\mathcal{P}$ was increased. This is to be expected, since more of the solution is being computed via direct matrix methods rather than Picard iteration.

- The theory of the SOR scheme states that there is an optimal value of ω that leads to the quickest convergence of the algorithm [136]. Our eigenvalue results showed that as ω is increased from $\omega = 1$ to some maximum value depending upon N, the spectral radius of the Jacobian matrix decreases. The smaller the spectral radius, the faster the rate of convergence of the SOR scheme. Although we did not attempt to find the optimal value of ω for this problem, our eigenvalue information in theory could be used for this purpose (by trial and error).

These behaviors are what one would expect from a correctly implemented Newton-Picard algorithm. From this and the accuracy of the eigenvalue and solution computations, we concluded that the basic framework for the code is correct.

5.2.2 Incorporating the TAY Program

Once the computational structure around the TAY code was developed, we conducted many types of tests to ensure the correctness of the code. We describe these tests in the next few sections. First, we explain how the accuracy and self-consistency of the eigenvalue computations were verified. Next, we show that the calculations made by the NPTAY program are consistent with those produced by TAY for those computations both codes can perform. This includes both equilibrium and time-periodic solutions. Finally, we compare some of our bifurcation computations with other numerical experiments.

Jacobian-Vector Products and Eigenvalue Calculations

To test our eigenvalue solver and implementation of Jacobian-vector products, we considered the cylinder geometry $\eta = 0.615$, $\Gamma = 2.4$. Using the TAY code, we computed the equilibrium solution at $(R_i, R_o) = (300, -80)$ on an 11×25 rz-grid ($\Delta r = \Delta z = 0.1$). Denote this solution by the vector $\mathbf{u}$. It will serve as our base solution. With this grid size, the Jacobian matrix $\mathbf{F_u}$ has size 621×621.

The test used ARPACK to compute the dominant 15 eigenvalues of this Jacobian matrix. These computations were done four times, by the following methods:

- Using only the multiplicative action of the Jacobian matrix, but not constructing it explicitly. There are two ways to accomplish this:
  1. Use finite difference equation (5.5). We will call this scheme AFD (for Jacobian Action via Finite Differences).
  2. Use the linearized Navier-Stokes solver. We will call this scheme ALNS (for Jacobian Action via Linearized Navier-Stokes solver).
- Explicitly constructing the Jacobian matrix by computing its action on the standard basis vectors $\mathbf{e}_i$. Its multiplicative action thereafter is computed via

direct matrix-vector multiplication. As above, there are two variants for this method:

1. Use finite difference equation (5.5) for the multiplicative action on the $\mathbf{e}_i$. We will call this scheme EFD (for Explicit Jacobian via Finite Differences).
2. Use the linearized Navier-Stokes solver for the multiplicative action on the $\mathbf{e}_i$. We will call this scheme ELNS (for Explicit Jacobian via Linearized Navier-Stokes solver).

Figure 5.1 shows the dominant 16 eigenvalues computed by these four methods. The eigenvalues are plotted in the complex plane, with the horizontal axis being the real part and the vertical axis being the imaginary part, as usual. There is good agreement between the four methods. Note that the axis scales are quite small, reflecting the eigenvalue clustering mentioned earlier.

As a further check on the ARPACK eigenvalue solver, we used the MATLAB `eig` command on the matrix produced by the ELNS scheme to compute the full spectrum of the Jacobian. Figure 5.2 shows the resulting eigenvalues. Note the clustering near 1. Figure 5.3 compares the dominant 16 eigenvalues returned by ELNS using ARPACK and those by MATLAB. There is excellent agreement between the results of ARPACK and of MATLAB.

Parameter Continuation Problems

The next phase of the verification involved testing NPTAY in a parameter continuation context, using one of the Reynolds numbers as the parameter. Three different Taylor-Couette cylinder geometries were considered for this phase of the verification. The basic procedure in all three problems was to begin by computing a starting solution by the TAY code. Next, we changed one of the Reynolds numbers (inner or outer) in a regular way; for instance, by successively increasing it by one, or successively decreasing it by two. We then used NPTAY to compute the solution at the

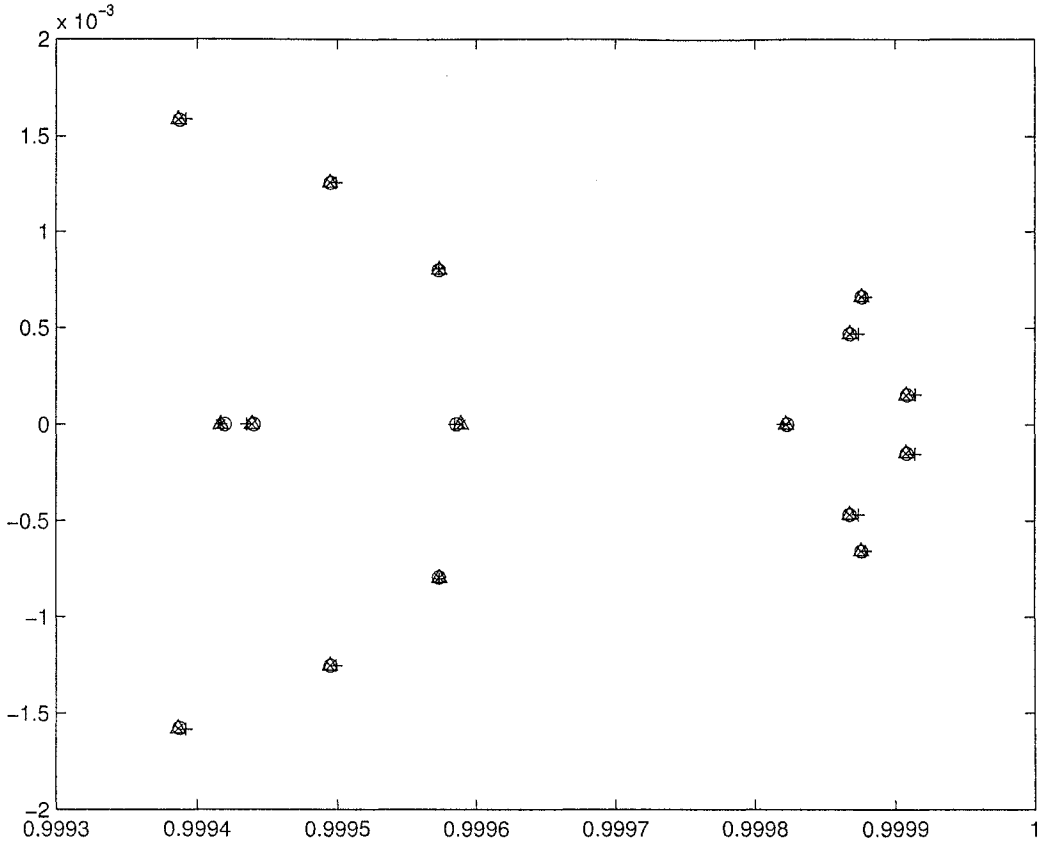

Figure 5.1: *Comparison of dominant eigenvalue calculations from the four methods for Jacobian-vector products. Eigenvalues are plotted in the complex plane; the horizontal axis represents the real part of the eigenvalue, and the vertical axis the imaginary part. The schemes used are: ELNS (+); EFD (×); ALNS (○); AFD (△).*

new Reynolds number, using the solution at the previous Reynolds number as the starting solution for the new Reynolds number. These solutions were compared with the corresponding solutions from the TAY code.

In all three cylinder geometries studied, a basis size of 15 and a convergence tolerance of 10^{-5} were used. One thousand Picard iterations were used per Newton step; that is, one thousand terms of the Neumann series for $(V_q^T M^\nu V_q - I)^{-1}$. The three problems are:

1. $\eta = 0.615$, $\Gamma = 2.4$ on an 11×25 $(r \times z)$ grid. The starting solution was at $(R_i, R_o) = (300, -80)$. The outer Reynolds number was decreased from

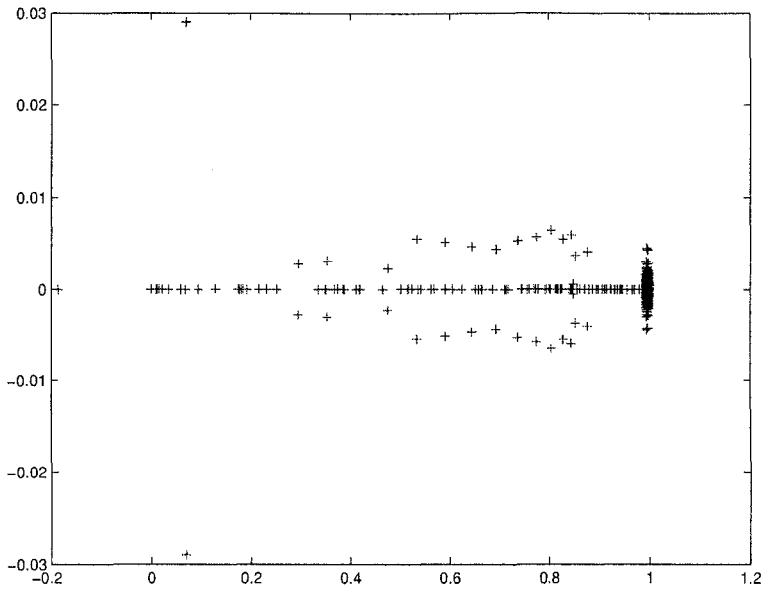

Figure 5.2: *The full spectrum of the Jacobian matrix as computed by* MATLAB, *plotted in the complex plane.*

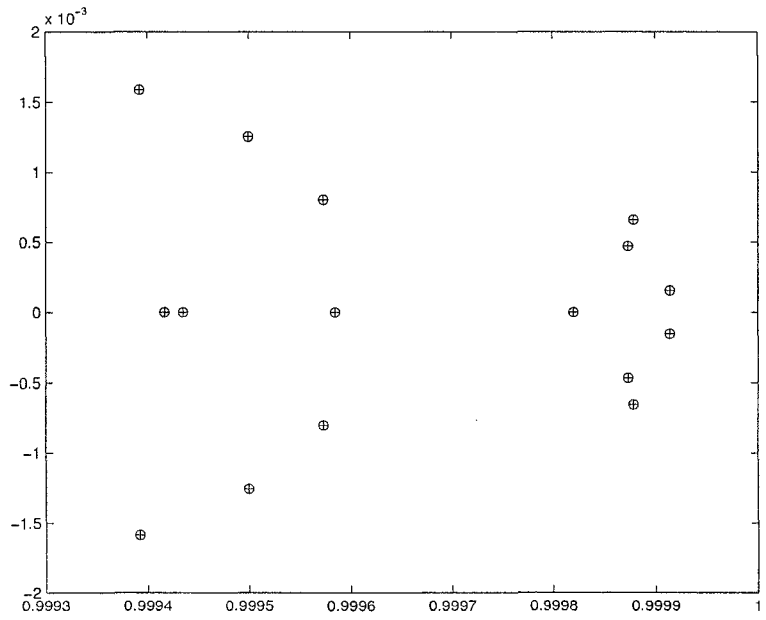

Figure 5.3: *Comparison of dominant eigenvalues returned by the ELNS scheme utilizing* ARPACK, *denoted by* $\circ$, *and those computed by* MATLAB, *denoted by* $+$.

$R_o = -80$ to $R_o - 88$ in increments of one while the inner Reynolds number was fixed at $R_i = 300$. The results are summarized in table 5.1.

2. $\eta = 0.5$, $\Gamma = 1.0$ on a 21×21 $(r \times z)$ grid $(\Delta r = \Delta z = 0.05)$. Starting at the solution at $(R_i, R_o) = (120, 0)$, the inner Reynolds number was increased from $R_i = 120$ to $R_i = 128$ in increments of two while the outer Reynolds number was fixed at $R_o = 0$ (stationary outer cylinder). The results are summarized in table 5.2.

3. $\eta = \Gamma = 0.7$ on a 21×15 $(r \times z)$ grid $(\Delta r = \Delta z = 0.05)$. The starting solution was at $(R_i, R_o) = (250, 0)$ and the inner Reynolds number was increased from $R_i = 250$ to $R_i = 258$ in increments of two while the outer Reynolds number was fixed at $R_o = 0$. The results are summarized in table 5.3.

For each of the summary tables, the second column gives the residual norm, which is a measure of how close the solution is to being a fixed point of the time-stepper map **F**. The third column gives the norm of the difference between the NPTAY solution and the TAY solution at that (R_i, R_o). All norms are sup-norms. As we can see, in all three test problems there is excellent agreement between the solutions computed by NPTAY and those generated by TAY.

Computation of the Period of Time-Periodic Solutions

The cylinder geometry $\eta = 0.615$, $\Gamma = 2.4$ on an 11×25 grid was also used to test the NPTAY code's ability to calculate the period of a time-periodic solution. For this cylinder geometry, the solution at $(R_i, R_o) = (300, -91)$ is periodic. We can see this by looking at time series plots (generated by TAY) of the azimuthal velocity at one particular gridpoint, in this case at $(r, \theta, z) = (0.3, 0, 1.1)$. This gridpoint was chosen because it is near the middle of the spatial domain, and we suspected any time-dependent behavior would be clearly visible in that area. The time series plot is given in figure 5.4; we estimate that the period is approximately 0.133, in terms of non-dimensional time.

(R_i, R_o)	*Residual Norm* $\|\mathbf{F}(\mathbf{u}) - \mathbf{u}\|$	*Solution Difference Norm*
$(300, -81)$	8.4×10^{-9}	1.55×10^{-5}
$(300, -82)$	5.3×10^{-9}	9.24×10^{-6}
$(300, -83)$	5.0×10^{-9}	9.47×10^{-6}
$(300, -84)$	5.5×10^{-9}	9.92×10^{-6}
$(300, -85)$	6.0×10^{-9}	1.03×10^{-5}
$(300, -86)$	2.3×10^{-9}	8.91×10^{-6}
$(300, -87)$	1.5×10^{-9}	3.98×10^{-6}
$(300, -88)$	2.0×10^{-9}	2.46×10^{-5}

Table 5.1: *Comparison of Newton-Picard solutions and those directly from the* TAY *code. The geometric parameters are* $\eta = 0.615$ *and* $\Gamma = 2.4$ *on an* 11×25 *grid.*

(R_i, R_o)	*Residual Norm* $\|\mathbf{F}(\mathbf{u}) - \mathbf{u}\|$	*Solution Difference Norm*
$(122, 0)$	1.34×10^{-9}	1.36×10^{-6}
$(124, 0)$	1.31×10^{-9}	1.35×10^{-6}
$(126, 0)$	1.27×10^{-9}	1.33×10^{-6}
$(128, 0)$	1.23×10^{-9}	1.32×10^{-6}

Table 5.2: *Comparison of Newton-Picard solutions and those directly from the* TAY *code. The geometric parameters are* $\eta = 0.5$ *and* $\Gamma = 1.0$ *on a* 21×21 *grid.*

(R_i, R_o)	*Residual Norm* $\|\mathbf{F}(\mathbf{u}) - \mathbf{u}\|$	*Solution Difference Norm*
$(252, 0)$	1.12×10^{-10}	7.84×10^{-7}
$(254, 0)$	2.62×10^{-10}	2.15×10^{-6}
$(256, 0)$	2.02×10^{-10}	2.15×10^{-6}
$(258, 0)$	2.65×10^{-10}	3.59×10^{-6}

Table 5.3: *Comparison of Newton-Picard solutions and those directly from the* TAY *code. The geometric parameters are* $\eta = \Gamma = 0.7$ *on a* 21×15 *grid.*

To compute the period of this solution using the Newton-Picard method, the solution from TAY at an arbitrary time was used as the starting solution for NPTAY. The Reynolds number was not changed and NPTAY was run to see what period it would compute for the solution. Also, because of the implicit phase condition imposed by solving the $\mathcal{P}$-equations (3.16) by a least squares method, the solution vector returned

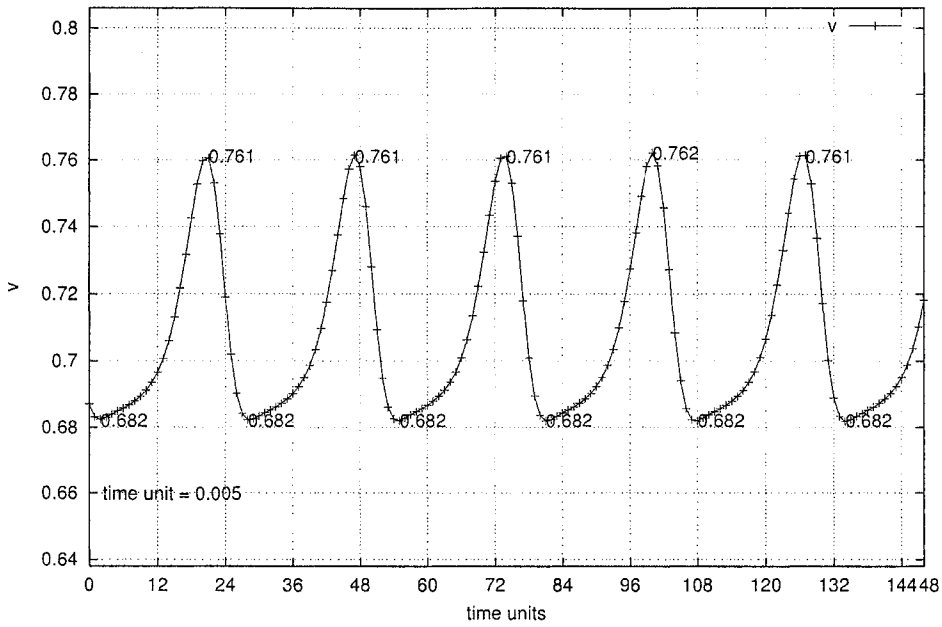

Figure 5.4: *Azimuthal velocity time series plot at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solution at* $R_i = 300$, $R_o = -91$.

by NPTAY would probably be different that the starting solution vector (but would still be a point on the periodic orbit). As determined by NPTAY, the period of the solution is 0.132, which is consistent with the period derived from the time series plot.

Comparison With Other Bifurcation Results

The final phase of verification involved using the eigenvalue information returned by NPTAY to determine if we could compute known bifurcation points that appear in the literature. We chose two Taylor-Couette geometries for the tests.

The first bifurcation test was based on the problem studied by Mullin, Toya, and Tavener [105], who looked at Taylor-Couette flows primarily for the cylinder geometry $\eta = 0.7$. In their experiments (both numerical and physical), they considered the location of symmetry-breaking bifurcation points as a function of both the inner Reynolds number (with $R_o = 0$ fixed) and the aspect ratio Γ. Because of how the time-stepper TAY was written, we cannot vary Γ in our problem. Hence, to compare

our results with theirs, we chose a value of $\Gamma = 0.7$ and considered the bifurcation which they suggested should occur somewhere near $R_i = 260$ (see Figure 3 of [105]). Using a 21×15 discretization of the rz-plane, our eigenvalue calculations showed that a single real eigenvalue crossed the line $\Re e(z) = 1$ between $R_i = 263.5$ and $R_i = 264$. This indicates that a bifurcation occurs between $R_i = 263.5$ and 264, which is consistent with the observations of [105].

The second test is based on the problem studied by Schulz, Pfister, and Tavener [119] and by Tavener [130]. We focused on the particular cylinder geometry $\eta = 0.5$ and $\Gamma = 1.0$. Their work suggests that a bifurcation should occur near $R_i = 132$ or $R_i = 133$ with $R_o = 0$ fixed. Using a 21×21 grid in the rz-plane, our eigenvalue results showed that a single real eigenvalue crossed the line $\Re e(z) = 1$ between $R_i = 132.6$ and $R_i = 132.8$, confirming the presence of a bifurcation there.

5.3 Summary

In this chapter, we have given details of the NPTAY code implementation and have validated the code for accuracy. We have highlighted the changes made to the original Newton-Picard method and to the original TAY program to successfully wrap the code around the time-stepper. Of particular interest is the fact that part of the time-stepper was rewritten because it was too slow to form a practical bifurcation code. This slowdown was caused by the use of a Gauss-Seidel iterative solver to solve the discrete Poisson equation. Using methods detailed in Appendix C, this solver was rewritten so that a single matrix-vector product gives the solution to the discrete Poisson solver. Pseudo-code algorithms of the method, as implemented in the final NPTAY code, also have been presented.

Details of the various tests used to determine the accuracy and correctness of the NPTAY code have been given. Tests involving the SOR algorithm, the ARPACK eigenvalue solver, parameter continuation, and period computations have all been performed and show that the NPTAY code produces accurate results. Known bifurcations in the Taylor-Couette problem also have been resolved by the NPTAY code.

Chapter 6

Numerical Experiments

6.1 Introduction

In this chapter, we describe the numerical experiments performed with the NPTAY code and give some of the bifurcation results obtained. The Reynolds numbers—in particular, the outer Reynolds number—play the role of the bifurcation parameter in all of the experiments. The structure of the underlying TAY code is such that the geometric parameters of the problem, namely the radius and aspect ratios, cannot be changed within an experiment. Since the NPTAY code is implemented around TAY, this structure therefore is imposed on NPTAY as well. Using techniques of numerical bifurcation theory, many numerical experimenters have been able to compute bifurcation curves where the Reynolds numbers, radius ratio, and aspect ratios all can play the role of the bifurcation parameter (see, for instance, [31], [108], [131], or [119] and the references therein). However, this ability is beyond the NPTAY code's capacity.

Keeping that in mind, a typical numerical experiment with the NPTAY code is performed as follows. An initial solution is generated using the TAY code. Sufficient time is allowed to ensure that the solution has reached equilibrium. Next, one of the Reynolds numbers is changed slightly, usually by one or two percent. This initial solution is used as the starting guess for the NPTAY algorithm to compute the solution at the new Reynolds number. This process is repeated, changing one of the Reynolds

numbers slightly and using the previously computed solution as the starting guess for the new Reynolds number. Bifurcations can be detected by monitoring the dominant eigenvalues of the Jacobian matrix, which routinely are computed as part of the Newton-Picard algorithm.

When the above procedure fails to converge, and after decrements in the Reynolds number step size also failed to give convergence, a turning point may be present. This hypothesis may be tested by monitoring the dominant eigenvalues and determining whether a single real eigenvalue approaches $+1$. When a turning point is suspected, the NPTAY code is run in arclength continuation mode. The numerical procedure for arclength continuation is similar to that described in the previous paragraph, but rather than changing the Reynolds number, the arclength parameter s is varied. Two initial solutions are generated by NPTAY, far from the suspected turning point. "Far" in our experiments usually means a Reynolds number difference of about 10%. Of these two initial solutions, the one closer to the turning point, as measured by the Reynolds number, is used as the solution at $s = 0$. The arclength continuation mode of NPTAY, as described in Section 3.4, then computes the solution at $s = \Delta s$, $s = 2\Delta s$, etc., until it is clear that the program has successfully traversed the turning point. This is done by observing the corresponding Reynolds numbers of the solutions and monitoring the dominant eigenvalues.

The ability of NPTAY to switch from one solution branch to another is limited and the process must be done manually. If a bifurcation on a stable solution branch is detected, rather than following the bifurcating solution branch, NPTAY will automatically continue on the original branch and compute solutions along this branch. To switch to the bifurcating stable branch, the user must generate a starting solution on the branch using TAY and then use NPTAY to continue along that branch. Switching to an unstable branch currently is not possible using this approach. For example, the user has little control over which unstable branch the code follows if a bifurcation

point on an unstable branch gives rise to even more unstable solution branches. We give a specific example of this later in the chapter.

In all of the numerical experimental results presented in this chapter, the aspect ratio is fixed at 2.4 and the radius ratio is 0.615. This is the same cylinder geometry explored by Adair and Thomas, as described in Chapter 4. All flows were assumed to be axisymmetric. The inner Reynolds number was held constant at $R_i = 300$ (except for one experiment), while the outer Reynolds number was varied over the interval $-320 \leq R_o \leq 0$. As described in Chapter 4, this region of the R_iR_o-parameter plane is ripe with many interesting behaviors, and we felt it worthwhile to explore it with the NPTAY code. All experiments, except those noted below, used a 10×24 rz-grid, resulting in the spatial discretization $\Delta r = \Delta z = 0.1$. A Newton iteration convergence tolerance of 10^{-5} was used, and the size of the basis of the invariant subspace $\mathcal{P}$ was chosen to be 15.

6.2 Two-Cell Flows Starting at $R_i = 300, R_o = -80$

First, we describe the variety of flows observed by using a starting solution at $R_i = 300, R_o = -80$. This solution was generated using TAY by starting at $R_i = 1, R_o = 0$, gradually increasing the inner Reynolds number to $R_i = 300$, and then gradually decreasing the outer Reynolds number to $R_o = -80$. The solution at $R_o = -80$ was computed after 800,000 time steps had elapsed to ensure sufficient equilibration time. The resulting flow is a stable, steady two-cell Taylor vortex flow; figure 4.1 in Chapter 4 is a velocity cross-section plot of the solution at $R_o = -80$.

As the value of R_o was decreased further from -80 to -91, a Hopf bifurcation was detected at $R_o = -90.6$, where the steady two-cell Taylor vortex flow loses stability to a time-periodic solution. This Hopf bifurcation was posited by Adair and Thomas in their numerical experiments described in Section 4.2.1.

Recall from equation (2.8) that an estimate of the initial period on a branch of time-periodic solutions emanating from a Hopf bifurcation is

$$T \approx \frac{2\pi\Delta t}{\omega}, \tag{6.1}$$

where ω is the imaginary part of the pair of eigenvalues crossing the line $\Re e(z) = 1$. In this particular case, we have $\omega = 0.002388$ and $\Delta t = 0.00005$, so that $T \approx 0.13156$ in terms of non-dimensional time. At $R_o = -91$, the period computed by NPTAY is 0.132, in very close agreement with this approximation. Further confirmation of this periodic solution branch was discussed in Chapter 5—see figure 5.4 in Section 5.2.2, where an estimate of the period at $R_o = -91$ is 0.133, based on velocity time series plots.

There are now two solution branches that can be followed: the unstable steady two-cell flow and the stable time-periodic flow. Following the unstable two-cell solution branch by decreasing R_o further, we found that another complex conjugate pair of eigenvalues crossed the line $\Re e(z) = 1$ at $R_o = -97$. The unstable steady flow becomes doubly unstable at this Hopf bifurcation point, giving rise to an unstable branch of time-periodic solutions. A third pair of eigenvalues crossed at $R_o = -98.5$, indicating yet another Hopf bifurcation. However, since all of these solution branches are unstable, the NPTAY code could not switch branches to follow the time-periodic ones. Instead the code continued to compute steady solutions along this now triply unstable branch. Further decreasing R_o to -320 showed that no further bifurcations occur on this unstable two-cell solution branch.

Now consider the time-periodic solution branch emanating from the Hopf bifurcation point at $R_o = -90.6$. Following this branch, we found that a complex conjugate pair of Floquet multipliers crossed the unit circle at approximately $R_o = -92.5$. This implies that there is a torus bifurcation point there, and that the time-periodic solution branch loses stability to a branch of time-dependent solutions with two frequencies. Because this time-dependent solution branch emanating from the torus

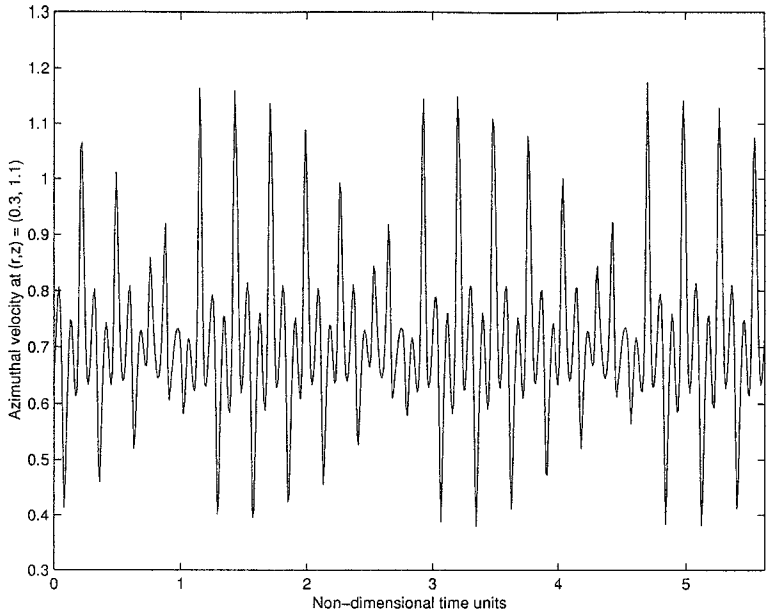

Figure 6.1: *Azimuthal velocity time series plot at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solution at* $R_o = -93$.

bifurcation point is stable, we can use the TAY code to obtain more information about it. Figure 6.1 shows a time series plot of the azimuthal velocity at a fixed gridpoint for $R_o = -93$, very near the torus bifurcation point. It is clear the solution is no longer periodic; rather, its time dependence is more complicated. It is difficult to discern by visual inspection how many independent frequencies appear in figure 6.1. Applying a discrete Fourier transform (DFT) to the data, using 560 data points, we obtained the power spectrum pictured in figure 6.2. It is evident that there are two well-defined peaks, indicating two incommensurate frequencies, which confirms the torus bifurcation.

Unfortunately, NPTAY cannot follow the branch of solutions emanating from the torus bifurcation point; it only can detect when this bifurcation occurs. The reason is that the solutions are not truly time-periodic (in exact arithmetic), and the Newton-Picard method is only applicable to periodic or equilibrium solutions. There is no analog of a Floquet multiplier for these multi-frequency solutions. However, state-of-the-art bifurcation software programs, such as the AUTO, CONTENT, and MATCONT

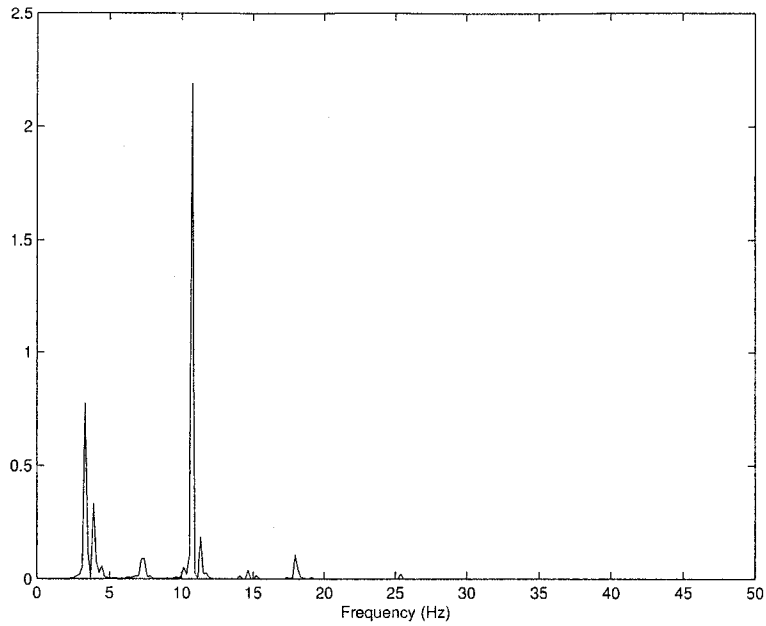

Figure 6.2: *Power spectrum via finite Fourier transform of the time series plot in figure 6.1.*

packages described in Chapter 1, cannot follow this solution branch either, for the same reasons. These software packages can only detect torus bifurcation points and, in the case of two parameter problems, compute paths of torus bifurcation points as these parameters are varied.

Therefore, if we wish to gather more information about this particular stable solution branch, the only recourse is to use temporal simulation via the TAY code. Figure 6.3 illustrates the solution behavior along this branch at $R_o = -94, -96, -98,$ and -100. These are time series plots of the azimuthal velocity at the same gridpoint as in figure 6.1. At $R_o = -94$, one can see more clearly that there are two frequencies present.

It is interesting to note that there is a change in the solution behavior between $R_o = -96$ and $R_o = -98$. The time-dependence of the flow is significantly different at $R_o = -98$ than it is at $R_o = -96$. Again using DFTs, we can construct power spectra for the solutions. Figure 6.4 shows the power spectrum at $R_o = -96$ and

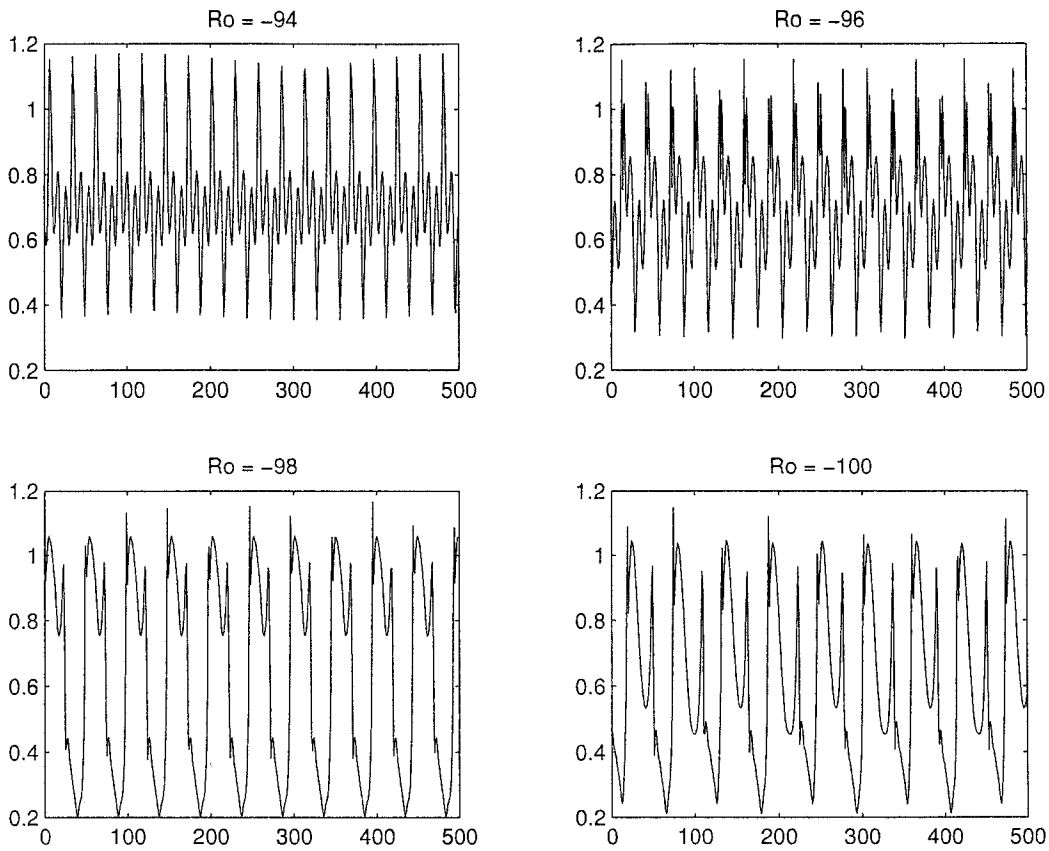

Figure 6.3: *Azimuthal velocity time series plots at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solutions at* $R_o = -94, -96, -98,$ *and* -100 *obtained by following the stable solution branch emanating from the torus bifurcation point at* $R_o = -92.5$. *The horizontal axis represents the number of time steps; each time step is* 0.01 *non-dimensional time units.*

$R_o = -100$. The solution at $R_o = -100$ still appears to have two independent frequencies; however, these frequencies are different from the ones computed for the solution at $R_o = -96$. The frequency near 10 Hz has been replaced by one near 2 Hz.

Although we could not follow the stable solution branch past the torus bifurcation point, we could follow the unstable time-periodic solution, something that, to our knowledge, no other Taylor-Couette code is able to do. Further decreasing the outer Reynolds number, at $R_o \approx -93$ we discovered an interesting phenomenon: the pair of Floquet multipliers lying outside the unit circle coalesced to a single real Floquet multiplier ≈ -1.5. This is interesting because if we had not been careful in detecting

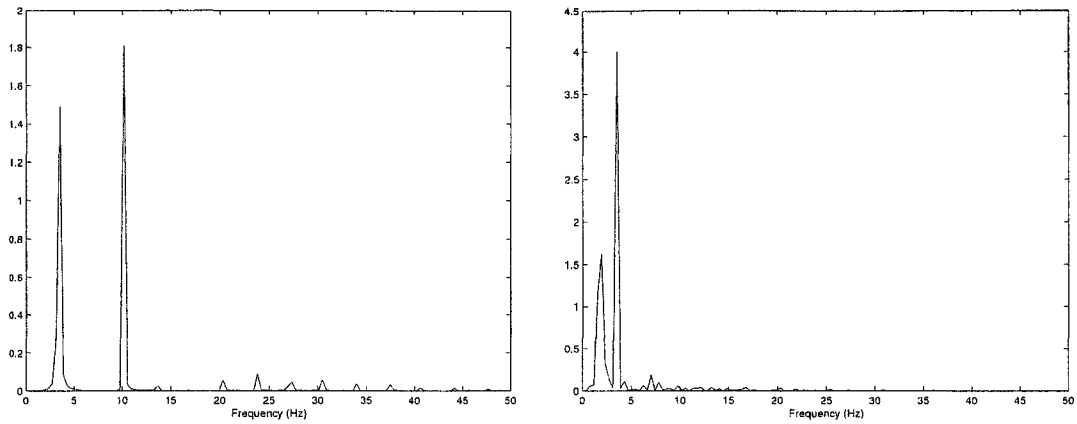

Figure 6.4: *Power spectra generated from the time series plots in figure 6.3 at* $R_o = -96$ *(left) and at* $R_o = -100$ *(right).*

the torus bifurcation at $R_o = -92.5$, we may have (and indeed initially did) come to the wrong conclusion. A time series plot of the solution at $R_o = -93$ is given in figure 6.5. The difference between that plot and the one given in figure 6.1 is that the latter uses nearly three times as many sample points as the former. A cursory look at figure 6.5 suggests that the solution at $R_o = -93$ is in fact time-periodic with period $T \approx 0.28$. The fact that 0.28 is slightly more than double the period 0.132 at $R_o = -91$, coupled with the coalesced Floquet multiplier at -1.5, led us initially to believe that a period-doubling bifurcation had occurred when in fact none had. (Recall that a period-doubling bifurcation occurs when a Floquet multiplier leaves the unit circle through -1.) This example clearly shows one of the disadvantages of temporal simulation: incorrect results can be arrived at by looking at plots of the resulting solutions. In this case, the time interval sampled was too short to accurately capture the two independent frequencies of the stable time-dependent solution at $R_o = -93$.

Further following the unstable time-periodic solution branch for decreasing R_o, we computed that a single real Floquet multiplier crossed the unit circle through -1 at $R_o \approx -95.6$. This indicates that the unstable time-periodic solution undergoes a period-doubling bifurcation, resulting in an unstable time-periodic solution with twice the period. The NPTAY code does not have the ability to switch to this new unstable

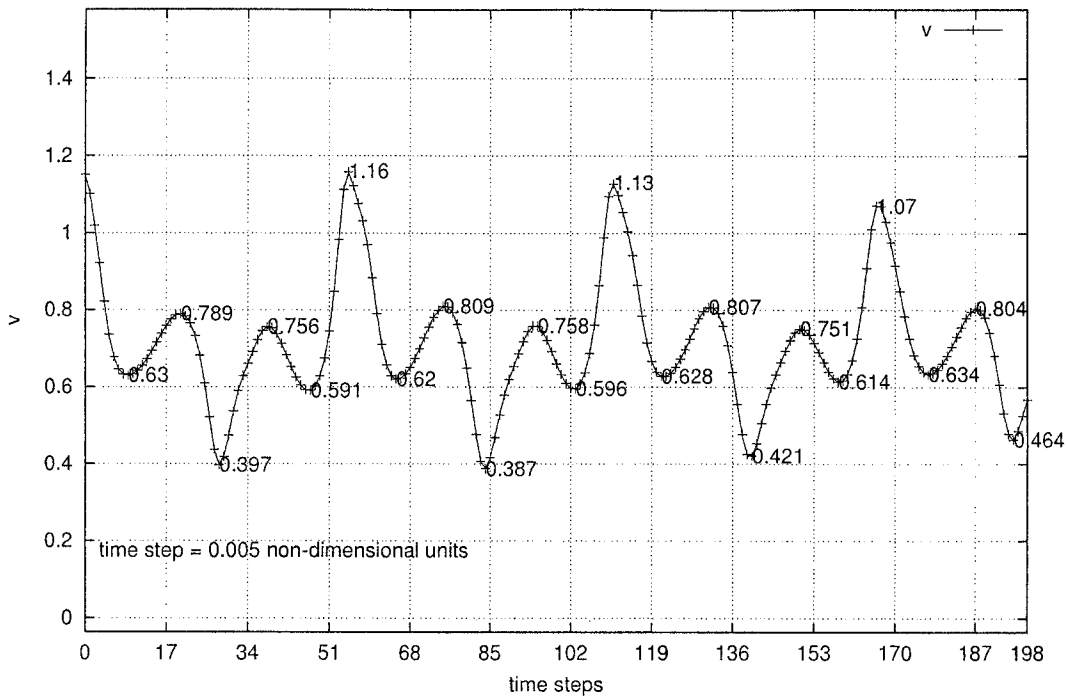

Figure 6.5: *Azimuthal velocity time series plot at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solution at* $R_o = -93$*, using one-third the number of sample points as figure 6.1.*

branch. Because the computational expense to find time-periodic solutions is much greater than that for equilibrium solutions, and because we cannot switch to any of these bifurcating solution branches, we chose not follow this branch any further. A summary of some of the results obtained in following the time-periodic branch are given in table 6.1.

6.3 Four-Cell Flows Starting at $R_i = 300, R_o = -320$

The next set of experiments use as a starting solution the stable steady four-cell Taylor vortex flow found at $R_i = 300, R_o = -320$. This flow enjoys a reflectional midplane symmetry about $z = 1.2$. Figure 4.2 is a velocity cross-section plot of this four-cell solution at $R_o = -320$.

R_o	*Stability*	*Period*	*Dominant Nontrivial Floquet Multipliers*
-91	stable	0.13224	$0.8429,\ -0.5738 \pm 0.4310i$
-91.3	stable	0.13264	$-0.6543 \pm 0.4874i,\ 0.7321$
-91.6	stable	0.13310	$-0.7364 \pm 0.5088i,\ 0.6122$
-91.9	stable	0.13353	$-0.8091 \pm 0.5034i,\ 0.4833$
-92.2	stable	0.13398	$-0.8791 \pm 0.4612i$
-92.5	unstable	0.13440	$-0.9439 \pm 0.3678i$
-92.8	unstable	0.13484	$-1.0067 \pm 0.0925i$
-93.1	unstable	0.13530	$-1.4923,\ -0.6526$
-95.2	unstable	0.13872	$-3.4534,\ -0.8271$
-95.5	unstable	0.13925	$-3.7559,\ -0.9599$
-95.8	doubly unstable	0.13980	$-4.0882,\ -1.0873$

Table 6.1: *Time-periodic solutions emanating from the Hopf point at* $R_o = -90.6$.

The experiment involves increasing R_o from -320. At $R_o = -189$, a single real eigenvalue of the Jacobian matrix became 1, indicating that this four-cell solution loses stability at that point. As we have mentioned before, an eigenvalue of unity generically indicates a turning point. Unless it is running in arclength continuation mode, the code should fail to converge near a turning point. However, the code did converge at $R_o = -189$, which suggests that something other than a turning point was detected. Because the four-cell flow possesses a reflectional symmetry about the midplane, it is plausible that a symmetry-breaking bifurcation occurs at $R_o = -189$. The group theoretic tools necessary to prove this were not incorporated into NPTAY. A more simple approach is to look at plots of the unstable and stable solutions at some point beyond the bifurcation point. Figures 6.6 and 6.7 show the unstable and stable solutions, respectively, at $R_o = -180$. A close inspection reveals that the unstable solution still possesses the midplane symmetry, whereas the stable solution shows a slight asymmetry, most visible near the midplane $z = 1.2$. This confirms the presence of a symmetry-breaking bifurcation point at $R_o = -189$.

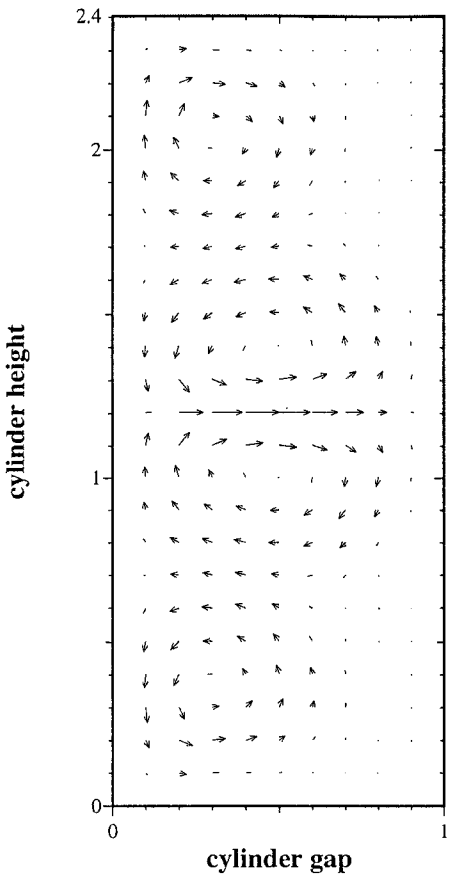

Figure 6.6: *Symmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -180$. *This equilibrium solution is unstable.*

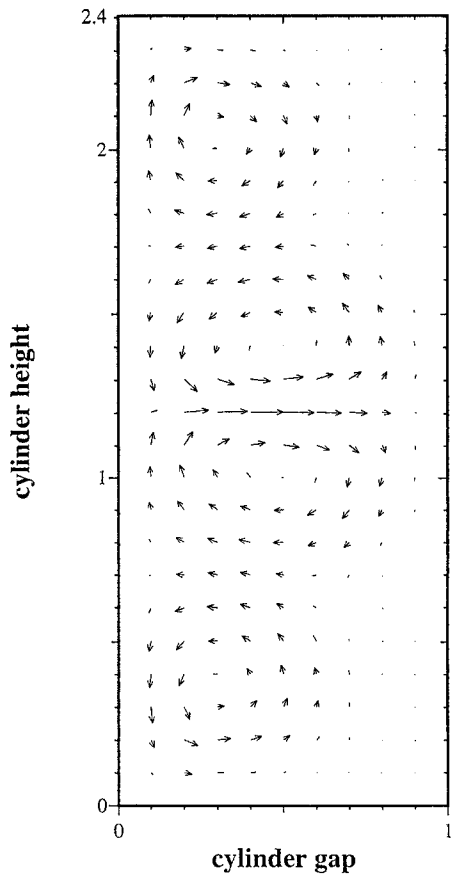

Figure 6.7: *Asymmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -180$. *This equilibrium solution is stable.*

In his experiments with this four-cell flow, Thomas [1] did not detect this symmetry-breaking bifurcation point. Both the stable and unstable solutions are four-cell flows, and on first inspection appear quite similar. The difference between them is that midplane symmetry still is enjoyed by the unstable solution but not by the stable solution. The asymmetry is quite subtle at first, and even far from the bifurcation point, it is not readily apparent to the naked eye. This illustrates another pitfall of direct simulation codes: visual inspection of plots may fail to identify some bifurcation points.

For parameter values $R_o > -189$, there are two solution branches to follow. First, we followed the stable asymmetric four-cell branch emanating from the symmetry-

breaking bifurcation point. Increasing R_o from -189 to -100 showed no further bifurcations. The velocity field shows that the flows are still four-cell flows, but the asymmetry has grown more pronounced, as shown in figure 6.8. However, near $R_o = -99$, the NPTAY code failed to converge; an inspection of the eigenvalues showed a single real eigenvalue close to 1. This suggested a turning point occurs near this parameter value. Running NPTAY in arclength continuation mode, we confirmed that a turning point occurs at $R_o = -99.25$. Figures 6.8 and 6.9 show the four-cell flows at $R_o = -101$. The former is the stable steady solution, whereas the latter is the unstable steady solution obtained after traversing the turning point. Both velocity plots are qualitatively similar, with some minor differences near the midplane $z = 1.2$.

Similarly, when we followed the branch of unstable symmetric four-cell flows from $R_o = -189$, we did not detect any bifurcation points for increasing R_o until $R_o = -34.1$, at which point a turning point was found. As we continued to follow this now doubly unstable symmetric four-cell branch around this turning point, another symmetry-breaking bifurcation point was detected, this time near $R_o = -39$. Emanating from the doubly unstable symmetric four-cell branch are two branches of doubly unstable asymmetric four-cell flows. Because the symmetric flow (before the symmetry-breaking point) and the two asymmetric flows (after the symmetry-breaking point) are both doubly unstable, this singularity was difficult to detect by simply monitoring eigenvalues—these solution branches all have two real eigenvalues greater than one. Inspecting the velocity vector plots on either side of this singularity clearly shows that a symmetry-breaking bifurcation occurs. See figure 6.10 for a representative velocity vector plot of the doubly unstable asymmetric flow. The velocity field for the solution on the other asymmetric branch can be obtained by a reflection through the midplane $z = 1.2$.

Notice the presence of extra vortices near the top and bottom of the cylinders in figure 6.10. There is a large vortex near the bottom and a much weaker vortex near the top. These vortices first appeared on velocity vector plots of the symmetric

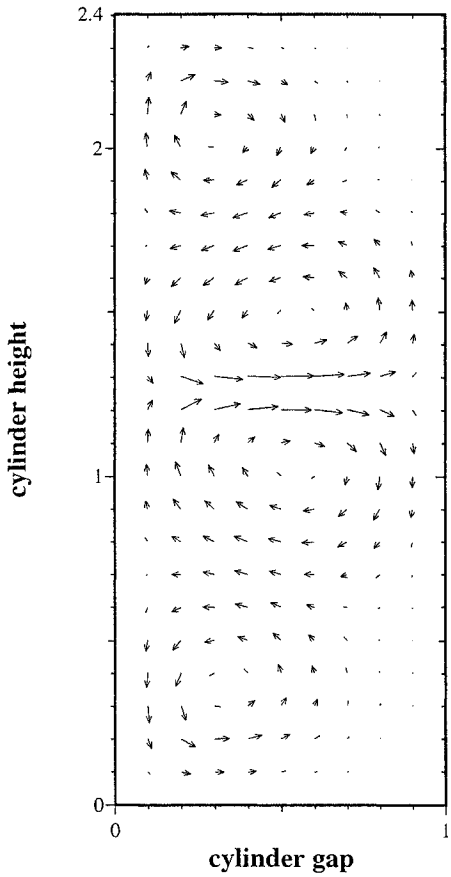

Figure 6.8: *Asymmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -101$. *This equilibrium solution is stable.*

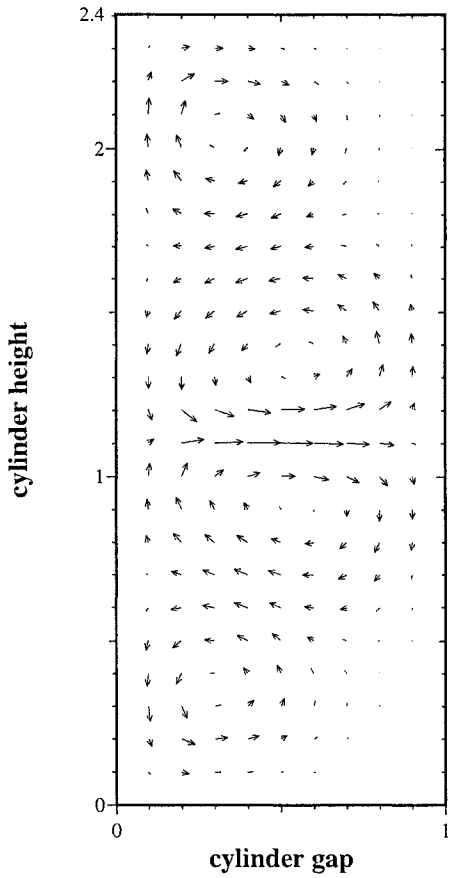

Figure 6.9: *Asymmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -101$ *after traversing the turning point at* $R_o = -99.25$. *This equilibrium solution is unstable.*

solution soon after the turning point at $R_o = -34.1$ was traversed. Although the presence of these extra vortices renders our nomenclature somewhat inaccurate, we still refer to all of these solutions as "four-cell" flows.

As R_o was decreased, the NPTAY code followed one of these asymmetric branches rather than following the now triply unstable symmetric branch. This is because user control of branch switching among unstable branches was not incorporated into the code. By varying the Reynolds number step size as the symmetry-breaking bifurcation point was approached, however, we were able to compute solutions on the symmetric four-cell branch past this symmetry-breaking bifurcation point. Figure 6.11 illustrates this triply unstable symmetric solution. Note that the extra vortices at both cylinder ends are quite pronounced.

To further confirm the evidence of a symmetry-breaking bifurcation point at $R_o = -39$, the axial velocity at the gridpoint $(r, z) = (0.3, 1.2)$ was computed for outer Reynolds numbers $-59 \leq R_o \leq -38$ on both asymmetric solution branches. Because this gridpoint lies on the midplane of the spatial domain, the axial velocity is zero for symmetric solutions and nonzero for asymmetric solutions. The resulting plot, given in figure 6.12, clearly shows the presence of a symmetry-breaking bifurcation point.

One final comment concerns one of the results obtained by Thomas [1]. As described in Section 4.2.1, using the original TAY code Thomas followed the branch of four-cell flows from $R_o = -320$ for increasing R_o, and discovered the presence of a time-dependent flow first appearing near $R_o = -120$. Our computations did not detect the presence of a Hopf bifurcation point there. Further experiments were unable to reproduce this result, using either TAY or NPTAY. More work needs to be done to explain this anomaly. However, the three-cell flow observed by Thomas when R_o was increased to -85 was analyzed by NPTAY, which we describe in the next section.

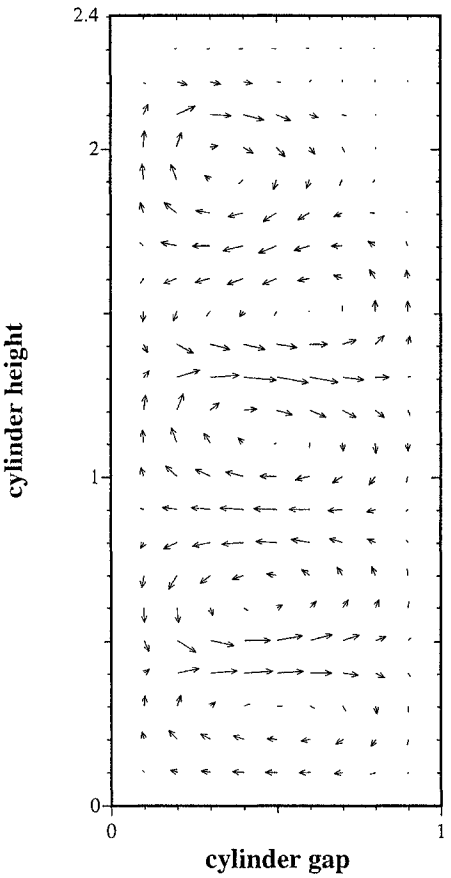

Figure 6.10: *Asymmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -59$. *This equilibrium solution is doubly unstable.*

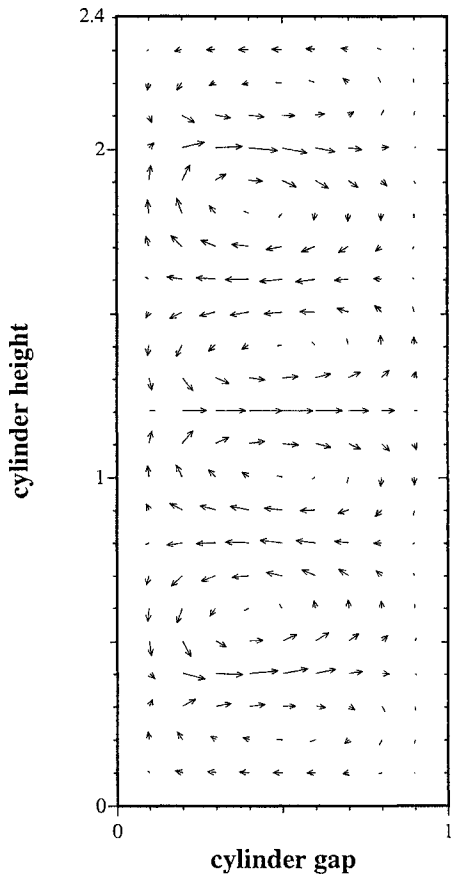

Figure 6.11: *Symmetric four-cell Taylor vortex flow velocity cross-section at* $R_o = -59$. *This equilibrium solution is triply unstable.*

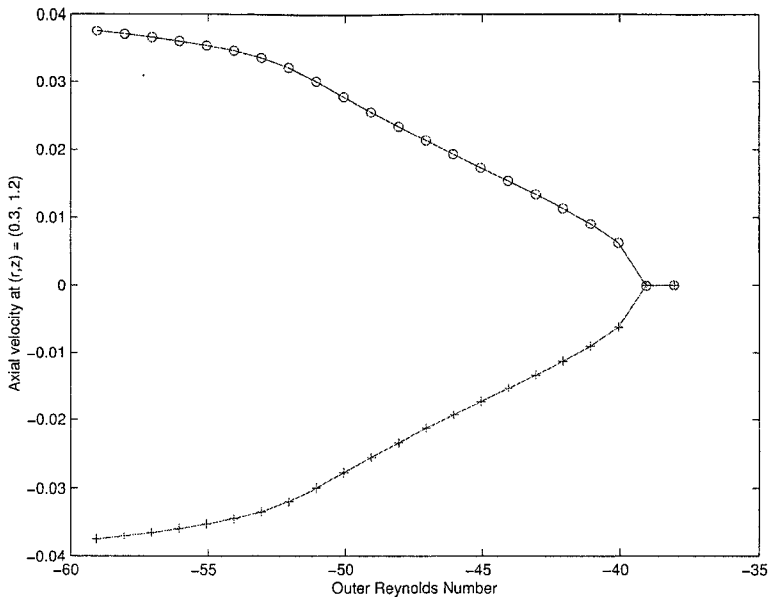

Figure 6.12: *Bifurcation diagram showing the presence of a symmetry-breaking bifurcation point at* $R_o = -39$.

6.4 Three-Cell Flows Starting at $R_i = 300, R_o = -85$

The final set of experiments used a three-cell Taylor vortex flow found at $R_i = 300, R_o = -85$ as a starting solution. Recall that we described the flows for $-90 \leq R_o \leq -80$ as being two-cell Taylor vortex flows. However, taking a more complicated path in the R_iR_o-parameter plane produced a three-cell Taylor vortex flow at $R_o = -85$. The path begins at $R_i = 1, R_o = 0$, goes to $R_i = 300, R_o = 0$, then to $R_i = 300, R_o = -320$, and finally to $R_i = 300, R_o = -85$. The resulting flow, after making sure to allow for sufficient settling time, is a three-cell flow which is steady and stable.

The procedure used to compute this three-cell flow illustrated yet another pitfall of time-stepping and provided an unanticipated additional benefit of the NPTAY code. Over three million time steps were required in TAY to obtain the three-cell solution at $R_o = -85$. Using either 500,000 or 1,000,000 time steps, the TAY code computed a four-cell flow at $R_o = -85$. However, an eigenvalue analysis using the

NPTAY code showed that these four-cell flows are in fact unstable, since both have one real eigenvalue greater than 1. One million time steps did not provide sufficient time at $R_o = -85$ to compute the correct stable solution. Without this eigenvalue data, we erroneously might have concluded that the four-cell flow was the stable flow at $R_o = -85$ along this path and that there was no three-cell flow at all. As described in Section 4.2.1, this was the initial conclusion arrived at by Thomas before he recomputed the solution using a much longer settling time. Thus, we see that eigenvalue calculations returned by NPTAY can provide insight into solutions returned by TAY and can help avoid this pitfall of not allowing sufficient settling time for the time-stepper to converge.

Beginning at $R_o = -85$, we decreased R_o to -320 and followed the three-cell solution branch along this path. At $R_o = -142.8$, there is a Hopf bifurcation. The imaginary part of the dominant eigenvalue at this point was approximately 0.0008. Since $\Delta t = 0.00005$, we can calculate from equation (6.1) that the initial period is approximately

$$T \approx \frac{2\pi\Delta t}{0.0008} = 0.393$$

in non-dimensional time. Figure 6.13 shows a time series plot of the azimuthal velocity at one gridpoint of the solution at $R_o = -143$. The solution is periodic with period $T \approx 0.405$, in close agreement with the estimate above.

Continuing to follow the (now unstable) three-cell solution branch past the Hopf bifurcation point, we found that the pair of eigenvalues that gave rise to the Hopf bifurcation return to the left of the line $\Re e(z) = 1$ at $R_o = -316$. Thus, the three-cell flow solution is re-stabilized at this point. Since we have followed the three-cell branch the entire length of this path, we know that the stable three-cell solutions for $R_o < -316$ and $R_o > -143$ are on the same branch. Direct simulation could not prove this. Using TAY to compute the stable three-cell solutions at $R_o = -140$ and at $R_o = -320$, for example, would show only that the flows are three-cell Taylor vortex flows. Based only on output from TAY, we could not conclude that the flows are

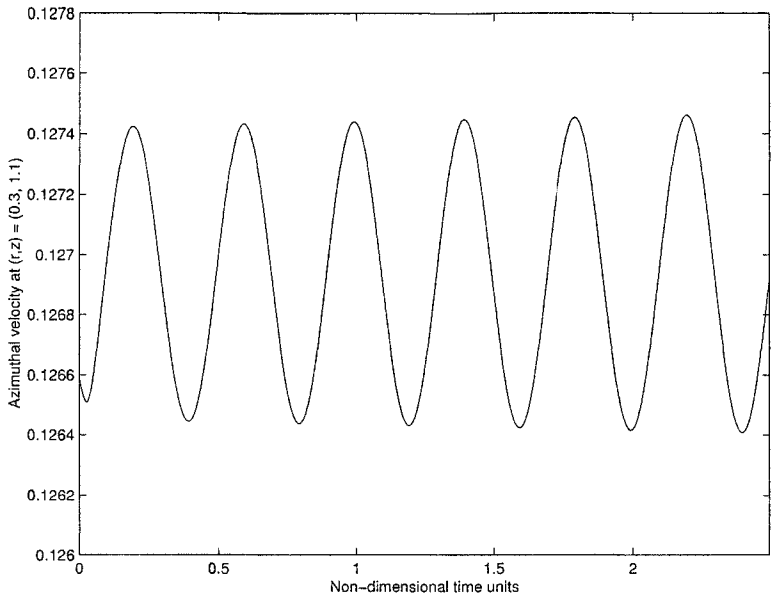

Figure 6.13: *Azimuthal velocity time series plot at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solution at* $R_o = -143$.

on the same solution branch. This is another illustration of the limitations of direct simulation.

In the next experiment, we increased the outer Reynolds number from -85 to 0. The NPTAY code detected a turning point at $R_o = -3.9$. Figures 6.14 and 6.15 show the stable and unstable three-cell flows, respectively, at $R_o = -8$. Notice the very weak additional vortex near the top corner of the cylinders in figure 6.15. This phenomenon of additional weak vortices is common in Taylor vortex flows with an odd number of vortices (see, for example, [8], [31], or [108]).

The results of the previous experiment contradict one of the results of Thomas described in Section 4.2.1. Thomas observed that the three-cell flow originating at $R_o = -85$ continues to persist to $R_o = 0$. Our calculations show that this cannot be possible, since there is a turning point at $R_o = -3.9$. Note, however, that Thomas used a finer spatial discretization than we did. His grid used $\Delta r = \Delta z = 0.05$ spacing, while ours was twice that. We conclude that either the coarser discretization

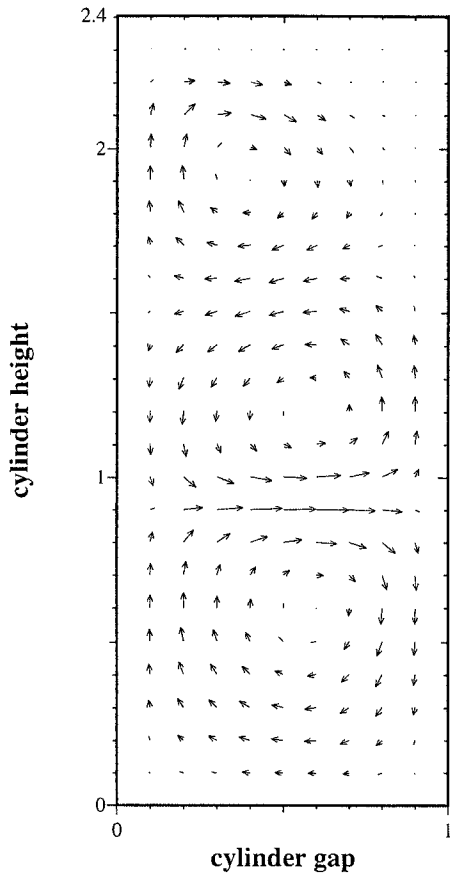

Figure 6.14: *Three-cell Taylor vortex flow velocity cross-section at* $R_o = -8$. *This equilibrium solution is stable.*

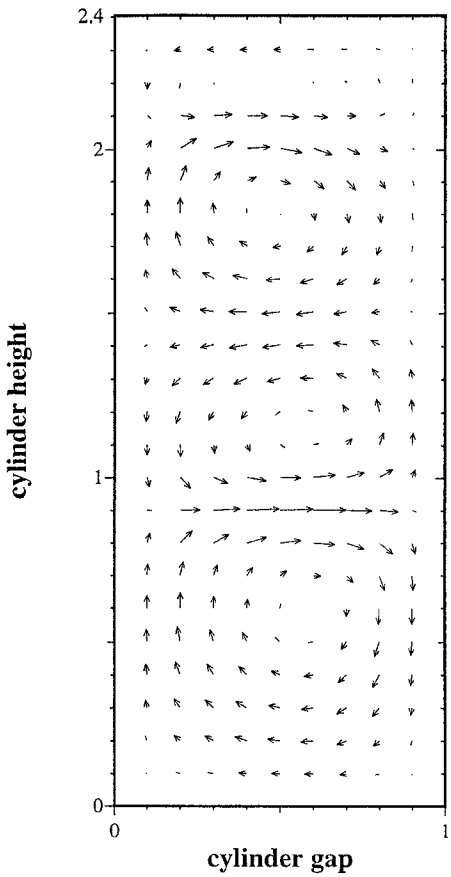

Figure 6.15: *Three-cell Taylor vortex flow velocity cross-section at* $R_o = -8$ *after traversing the turning point at* $R_o = -3.9$. *This equilibrium solution is unstable.*

introduced a turning point where none truly exists or the code detected it at a different value of outer Reynolds number.

To resolve this discrepancy, we recomputed all of our results using the finer discretization $\Delta r = \Delta z = 0.05$. Increasing the outer Reynolds number from -85, the NPTAY code confirmed the presence of a turning point near $R_o = 1$. Figures 6.16 and 6.17 show the stable and unstable solutions, respectively, at $R_o = 0.046$. As with the coarser grid, we see an additional weak vortex in the top corner of the unstable flow. However, this resolution allows us to see that this additional vortex also is present in the top corner of the stable solution, although it is less pronounced than in the

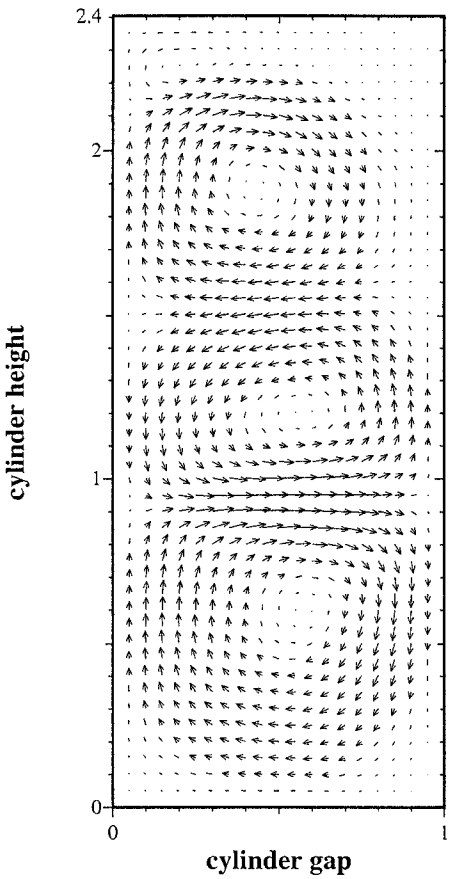

Figure 6.16: *Three-cell Taylor vortex flow velocity cross-section at* $R_o = 0.046$ *with spatial discretization* $\Delta r = \Delta z = 0.05$. *This equilibrium solution is stable.*

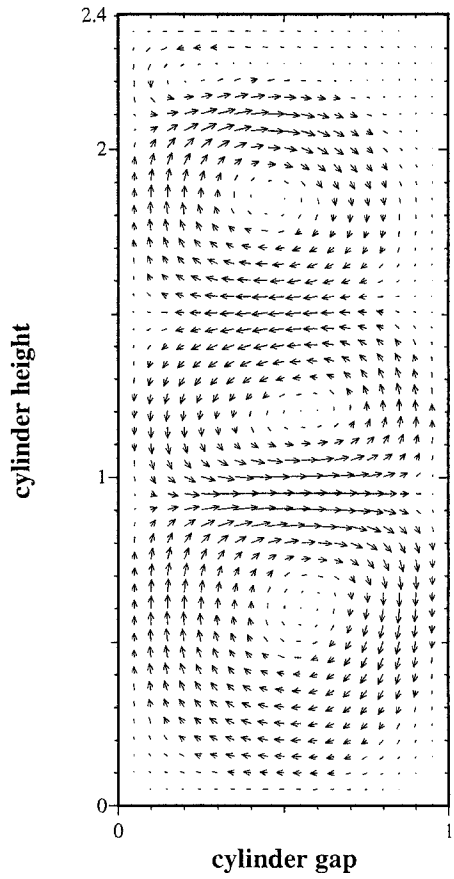

Figure 6.17: *Corresponding unstable three-cell Taylor vortex flow velocity cross-section at* $R_o = 0.046$, *after traversing the turning point at* $R_o = -3.9$.

unstable solution. As mentioned above, such weak vortices are common in Taylor flows with an odd number of vortices.

We conclude that using the coarser discretization did not qualitatively change the solution branch behavior. We have confirmed that there is a turning point, but the parameter value at which this occurs for the finer discretization is about five greater than that of the coarser discretization. This also resolves the discrepancy between our coarse grid results and Thomas' results at $R_o = 0$.

Our final computation involving the three-cell flow solution branch was the only one conducted where the inner Reynolds number was varied. Thomas observed that

the three-cell flow at $R_i = 300, R_o = 0$ suddenly gave way to a two-cell flow as R_i was decreased to $R_i = 295$. This abrupt transition led us to suspect that a turning point is present on the three-cell solution branch near $R_i = 295, R_o = 0$. Indeed, when run in arclength continuation mode, NPTAY confirmed the presence of this turning point at $R_i = 295.4$. Note that this computation was done on the finer 20×48 grid, since the three-cell solution branch does not exist at $R_o = 0$ when using the coarser grid.

6.5 Comparison with ENTWIFE

The bifurcation results presented in this chapter were compared with those of the finite-element code ENTWIFE [28]. In addition to being able to solve the axisymmetric Navier-Stokes equations, ENTWIFE also incorporates numerical bifurcation techniques, such as Keller's arclength continuation [73] and the extended systems developed by Moore and Spence [102] and Werner and Spence [145], that enable it to follow solution branches and compute curves of singular points.

Because it implements these sophistical numerical bifurcation techniques, ENTWIFE is very efficient. As a result, experiments can be performed with ENTWIFE using a much finer grid with many more degrees of freedom (total number of unknowns) than we are able with NPTAY. The total number of degrees of freedom used in these calculations was 10,212. Furthermore, ENTWIFE employs a corner refinement treatment, whereby the element grid is successively refined near all four cylinder-end interfaces and the velocity is interpolated so that it is continuous at these interfaces. Contrast this with the corner treatment by TAY, which leaves a velocity discontinuity at each corner. Most of the ENTWIFE calculations described in this section were performed by S. Tavener [personal communication].

6.5.1 Two-Cell Flows

Recall that on the branch of two-cell flows beginning at $R_o = -80$, NPTAY calculates that a Hopf bifurcation occurs near $R_o = -90.6$. This result is based on the

rather coarse 10×24 spatial discretization grid. For comparison with ENTWIFE, we chose to recompute the location of this Hopf bifurcation point using a finer grid. A 20×48 grid gives twice the resolution as the coarser grid, but at the cost of quadrupling the size of the system dimension. Using this finer grid, we calculated that the Hopf point occurs at $R_o = -93.2$. Because of the large computational expense involved with computing on this finer grid, we used this result as the basis for comparison with ENTWIFE, rather than attempting calculations on even finer grids.

Starting on the same solution branch, ENTWIFE calculated that the Hopf bifurcation occurs near $R_o = -98.5$. We are confident of this figure to three significant figures, as it is based on a grid refinement study within ENTWIFE. Thus, there is a 5.4% relative difference between the two results. Removing the corner refinement within ENTWIFE changed its result by approximately 2%, not enough to explain this difference in the location of the Hopf bifurcation point. Although there is a slight numerical difference between the results of NPTAY and ENTWIFE, qualitatively the bifurcation behavior is the same.

6.5.2 Three-Cell Flows

We next compared our results on the solution branch of three-cell flows with those of ENTWIFE. As in the two-cell flows above, we used as our reference the NPTAY results obtained on the finer 20×48 grid. In Section 6.4, all experiments but the Hopf bifurcation were repeated on this finer grid. Repeating the Hopf calculation on the finer grid, we found that the location of the Hopf point is approximately $R_o = -103$. The time series plot generated by TAY in figure 6.18 confirms that the solution is indeed time-periodic at $R_o = -107$ and its period is compatible with that obtained using equation (6.1). The solution at $R_o = -107$ was chosen because it is far enough from the location of the Hopf point that the TAY code could easily converge to the stable solution, but it is far enough from the Hopf point that equation (6.1) provides only a rough estimate of the solution's period.

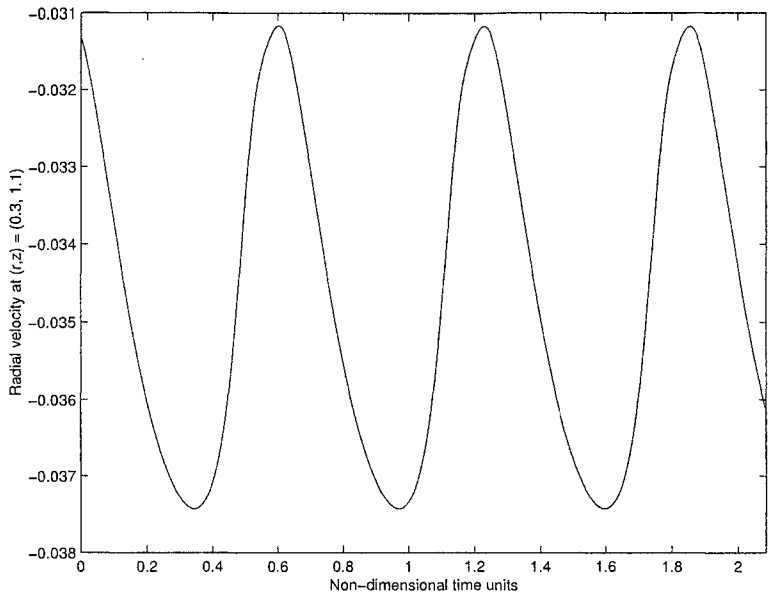

Figure 6.18: *Radial velocity time series plot at gridpoint* $(r, z) = (0.3, 1.1)$ *for the solution at* $R_o = -107$.

In summary, there are three singularities on the branch of three-cell solutions that we can compare with ENTWIFE: a Hopf bifurcation near $R_o = -103$, a turning point near $R_i = 300, R_o = 1$, and a turning point at $R_i = 295.4, R_o = 0$. The results returned by ENTWIFE confirm the presence of these two turning points. ENTWIFE calculated that there are turning points at $R_i = 300, R_o = 1.2$ and $R_i = 294.5, R_o = 0$, which are very close to our values.

However, ENTWIFE did not detect the Hopf bifurcation at $R_o = -103$ found by NPTAY. Instead, it calculated that there is a turning point near $R_o = -84$. Since this is near the location ($R_o = -85$) where our calculations on the three-cell branch began, we repeated the calculations starting with the stable three-cell solution returned by TAY at $R_o = -75$. The same behavior is seen when starting at $R_o = -75$ as at $R_o = -85$: as R_o is decreased, a Hopf bifurcation occurs near $R_o = -103$ and no turning point is detected at all along the branch.

To ensure that the NPTAY code was not missing the turning point seen by ENTWIFE inadvertently, the TAY code was used by itself, starting at the solution at

$R_o = -75$. The NPTAY code was used only to calculate dominant eigenvalues and hence determine the stability of the solutions returned by TAY. For decreasing R_o, TAY does not exhibit behavior that is consistent with a turning point being present at $R_o = -84$. Instead, stable three-cell flows are found for $-99 \leq R_o \leq -75$. As mentioned above, the flow is definitely periodic by $R_o = -107$, and the base flow upon which this time-periodicity is imposed is still a three-cell flow. Only slight variations in the velocity vector field are seen over its period. As shown in figure 6.18, the radial velocity at $(r, z) = (0.3, 1.1)$ varies by less than 0.007. We concluded that the results of these TAY experiments are consistent with those of NPTAY.

Note that the NPTAY results and those of ENTWIFE agree quite well when $|R_o|$ is small. Because of the increase in stress forces at the outer cylinder when $|R_o|$ is large, it is more difficult to resolve the fluid flow behavior at those Reynolds numbers accurately. Furthermore, the velocity discontinuity left by the TAY code at the outer cylinder becomes more pronounced for larger $|R_o|$. ENTWIFE attempts to ameliorate this by refining the element grid near those corners and smoothing the velocity over the elements adjacent to the outer cylinder. Thus, results from TAY are less likely to agree with those from ENTWIFE for large values of $|R_o|$.

6.5.3 Four-Cell Flows

The situation is less clear for the branch of four-cell flows starting at $R_i = 300, R_o = -320$. As with the three-cell flows, there is reasonable agreement between ENTWIFE and NPTAY for small $|R_o|$. In particular, using the finer 20×48 grid, NPTAY calculated a symmetry-breaking bifurcation point on the doubly unstable symmetric branch at $R_o = -43$. (Recall that this point occurs at $R_o = -39$ when using the coarser grid.) ENTWIFE found this symmetry-breaking bifurcation point at $R_o = -46$, which represents a 6.5% relative difference between the two results.

However, according to ENTWIFE calculations, the branches of four-cell flows are very complicated and folded. There are at least three branches of asymmetric four-cell

flows, on which occur at least four turning points. The full extent of the symmetric four-cell flows has not been fully studied using ENTWIFE, although six symmetry-breaking bifurcation points have been found so far. It would be extremely computationally expensive and time-consuming to try to resolve all details of the four-cell flows using the techniques available to the NPTAY code. We leave this as possible future work.

6.5.4 Conclusions

Many of the results given by NPTAY were confirmed by the finite-element code ENTWIFE. This is especially true for small values of $|R_o|$, which is not surprising since flows for small $|R_o|$ are more easily resolved than for larger $|R_o|$, and the velocity discontinuities present in TAY are not pronounced for small $|R_o|$. The two codes do disagree, however, on the presence of a turning point on the three-cell solution branch, but this occurs for a larger value of $|R_o|$. We found that the solution set of four-cell flows is much more complicated than initially suspected and we have not attempted to use the NPTAY code to resolve these flows fully because of the great computational expense required.

When comparing the results of these two codes, we should keep in mind the differences between NPTAY and ENTWIFE. First, the number of degrees of freedom used by the two codes in these experiments was quite different. By degrees of freedom we mean total number of solution unknowns, including all velocity values at all gridpoints. Using the finer 20×48 grid, the total number of degrees of freedom in NPTAY is 2,679. Compare this with ENTWIFE, where the total number of degrees of freedom is 10,212. This represents a significant difference in resolution between the two codes.

Second, the computational technique utilized by NPTAY is based on finite-difference approximations and finite Fourier series. ENTWIFE, on the other hand, uses a finite-element method to solve the Navier-Stokes equations. Finally, NPTAY introduces a

velocity discontinuity at the corner interfaces between the cylinders and the ends. ENTWIFE refines the element grid near these interfaces and interpolates the velocity so that it is continuous across the interfaces. Removing the corner refinement in ENTWIFE affected its results by only a few percent, however.

Thus, we see that the two codes, although solving the same governing equations, do so in different ways at different resolutions. Although the different resolutions and computational techniques should not matter by themselves, we hypothesize that the combined effects of these differences, the different treatments of the corner velocity discontinuities, and the inherent difficulty of accurately resolving fluid flows at larger Reynolds numbers, all contribute to the discrepancies observed. In the end, all of the bifurcation results obtained by either NPTAY or ENTWIFE only apply to the finite-dimensional dynamical system obtained after approximating the Navier-Stokes equations. Since these systems are quite different, it is not surprising that the two codes produce differing results. The results returned by NPTAY are internally consistent and are consistent with observed flow behavior as computed by TAY. Differences in solution behavior therefore must be attributed to the underlying solution schemes embodied in the Navier-Stokes solvers of TAY and ENTWIFE.

6.6 Summary

In this chapter we have described the numerical experiments conducted using the NPTAY code. The parameter region $R_i = 300$, $-320 \leq R_o \leq 0$ was explored by starting on three different steady solution branches: a two-cell branch, a four-cell branch, and a three-cell branch. We have confirmed Adair and Thomas' hypothesis that there is a Hopf bifurcation on the branch of two-cell flows near $R_o = -90$. We also observed a symmetry breaking bifurcation on the four-cell branch that was not detected by Thomas due to the very slight asymmetry produced. Also, we found several turning points, only one of which was suspected by Thomas.

Figures 6.19–6.21 give schematic diagrams for the various solution branches calculated, including any bifurcation and turning points. These figures are schematic representations only; there is no variable being measured in the vertical direction. Although many (but not all) of the computations described in this chapter were performed on two grids, one a refinement of the other, these figures depict the locations of the singularities as found using the coarser of the two grids. This was done for consistency, since computations involving time-periodic solutions are too expensive to be done on the finer grid and can only be performed on the coarse grid. Although the location of the singularities changed as the grid was refined, the qualitative behavior of the solution branches did not.

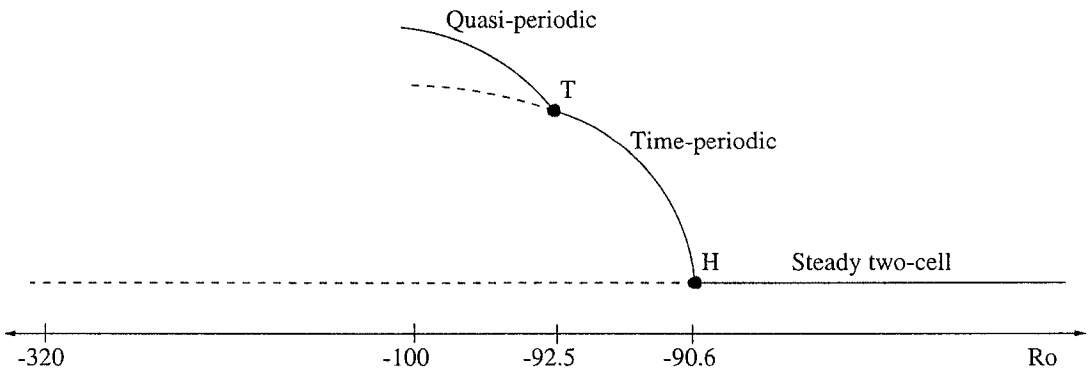

Figure 6.19: *Schematic bifurcation diagram for flows encountered starting at the two-cell flow at* $R_o = -80$. *Solid lines indicate stable solutions, dashed lines indicate unstable solutions. The point labelled H denotes a Hopf bifurcation point while the point labelled T denotes a torus bifurcation point.*

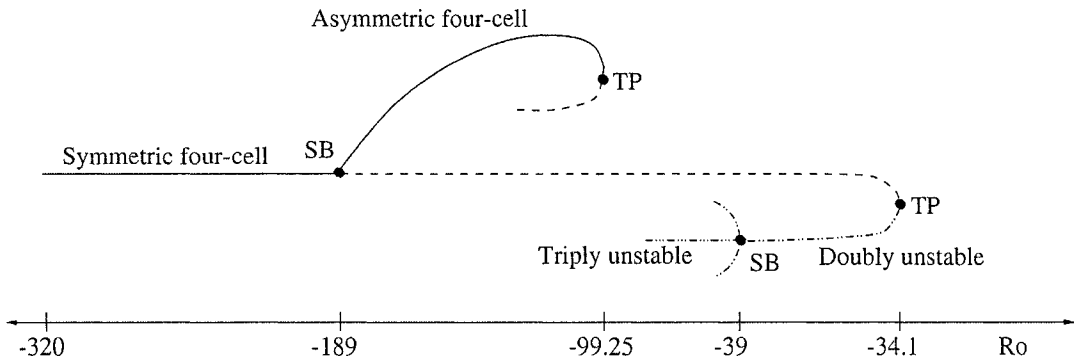

Figure 6.20: *Schematic bifurcation diagram for flows encountered starting at the four-cell flow at* $R_o = -320$. *Solid lines indicate stable solutions, dashed lines indicate unstable solutions, the dot-dot-dashed lines indicated doubly unstable solutions, and the dot-dot-dot-dashed line indicates a triply unstable solution. The points labelled* SB *denote symmetry-breaking bifurcation points and the points labelled* TP *denote turning points.*

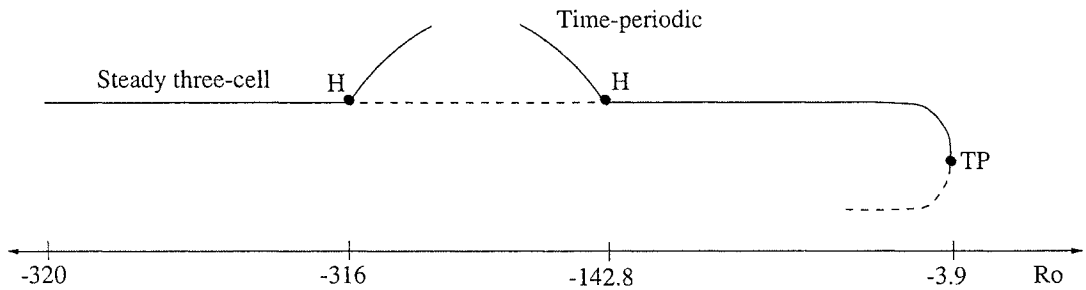

Figure 6.21: *Schematic bifurcation diagram for flows encountered starting at the three-cell flow at* $R_o = -85$. *Solid lines indicate stable solutions and dashed lines indicate unstable solutions. The points labelled* H *denote Hopf bifurcation points and the point labelled* TP *denotes a turning point. The turning point at* $R_i = 295.4$, $R_o = 0$ *is not illustrated in this diagram.*

Chapter 7

Conclusions and Future Work

In this chapter, we summarize the work presented in this dissertation and analyze the conclusions we can draw from it. We also give some directions for future work in this area.

7.1 Summary

As outlined in Chapter 1, our goal was to write a computational structure around an existing time-stepper code to perform a bifurcation analysis beyond the time-stepper's original capability. We have successfully done this for the code TAY written by R. Adair [2] for studying the Taylor-Couette problem. As with most time-steppers, the TAY code suffers from many limitations which make it unsuitable for performing a complete bifurcation study, including:

1. The convergence of the code becomes arbitrarily slow as a bifurcation point is approached. Many thousands, or even millions, of time steps can be required to compute the stable solution near such points. Even then, one cannot be certain whether the resulting solution truly is the stable solution or whether more time steps are needed to ensure a sufficient equilibration time. As a consequence of this, the code cannot locate bifurcation points accurately. Only an approximate location can be found, since qualitative changes in the solution may become apparent only farther away from the bifurcation point. Even more problematic,

bifurcations that produce only subtle changes in the solution appearance can be missed entirely. This happened to J. W. Thomas [1]. While computing four-cell Taylor vortex flows, a symmetry-breaking bifurcation point was overlooked because the resulting flow which looks remarkably like the solution before the bifurcation point.

2. The TAY code can compute only stable solutions; unstable solutions are beyond its capability.

3. The TAY code cannot compute the period of time-periodic solutions. Only an *a posteriori* estimate of the period can be made via time series plots.

4. The distinction between a turning point and a subcritical bifurcation point cannot be made by the TAY code. As either type of singularity is crossed, the TAY code converges to a stable solution on another branch. The changes observed in the solution behavior are very similar in both cases. In the case of a supercritical bifurcation point, addition information from TAY, such as velocity field plots, contour plots, or time series plots, often can distinguish a supercritical bifurcation point from a turning point.

The TAY code does possess one distinct advantage: it can compute and follow branches of very complicated time-dependent solutions. We observed this when a torus bifurcation point was encountered along a branch of time-periodic solutions. The emanating stable branch cannot be followed except by a time-stepper. Although not studied in this dissertation, even more complicated behavior can be computed with the TAY code.

Our code, which we call NPTAY, uses the Newton-Picard method of K. Lust [85] as the mathematical basis for the wrap-around algorithm. The method is an extension of the recursive projection method (RPM) of H. B. Keller and G. Shroff [121] which splits the problem solution into two parts. Part of the solution is computed

via a direct linear solver in a small-dimensional subspace corresponding to the most dangerous modes. The other part of the solution is computed in the large-dimensional orthogonal complement by an iterative method which is essentially the linearization of the iteration used in the TAY code. We used the Newton-Picard method, rather than the RPM, because of its robustness and ability to compute time-periodic solutions.

As a by-product of this splitting procedure, the eigenvalues of the linearized operator are computed. These eigenvalues give information about solution bifurcations, since eigenvalues crossing the line $\Re e(z) = 1$ indicate a bifurcation has occurred. A complex conjugate pair of eigenvalues crossing this line, for instance, indicates a Hopf bifurcation point, where the steady solution branch loses stability to a branch of time-periodic solutions. This eigenvalue information also can be used to determine the stability of a solution, which can be used in turn to calculate whether a solution from the TAY code has had sufficient settling time. We used this feature while computing the three-cell Taylor vortex flow at $R_i = 300, R_o = -85$.

Since the part of the solution corresponding to the most dangerous modes is computed by direct methods rather than by iteration, the convergence rate of NPTAY near bifurcation points is accelerated. Also, the code can compute unstable solutions, since the projection of the iteration onto the orthogonal complement of the dangerous eigenspace is still a contraction. Both unstable equilibrium and unstable time-periodic solution branches can be followed using our code. Finally, when coupled with arclength continuation, the NPTAY code can detect the location of turning points and then follow the solution branch around the turning point. The unstable solutions on the other side of the turning point can be computed also.

Using the NPTAY code, we conducted a bifurcation study in the region of parameter space studied by Adair and Thomas [1, 2]. In this region, the inner Reynolds number is fixed at $R_i = 300$ and the outer Reynolds number is varied over the interval $-320 \leq R_o \leq 0$. Our calculations are confined to axisymmetric flows—those that are

independent of the azimuthal variable θ. Depending on which solution branch we started on, we detected different bifurcations points and behaviors.

For instance, starting on a two-cell Taylor vortex flow solution at $R_o = -80$, we found a Hopf bifurcation point at $R_o = -90.6$. The resulting time-periodic solution has a period of about 0.13 non-dimensional time units as computed by our code. This nicely matches both an estimate of the period as determined by time series plots generated by TAY, and an estimate obtained by using the imaginary part of the dominant eigenvalue. Following this emanating time-periodic solution branch, we quickly discovered another bifurcation point. The time-periodic branch undergoes a torus bifurcation at $R_o \doteq -92.5$, wherein a complex conjugate pair of Floquet multipliers leaves the unit circle. The resulting flow cannot be analyzed by our code, but time series plots from TAY confirm the presence of two independent frequencies in the new flow.

When we instead began at a four-cell Taylor vortex flow at $R_o = -320$, we found a symmetry-breaking bifurcation point at $R_o = -189$. The resulting flow is still a four-cell flow, but one which does not possess the midplane reflectional symmetry enjoyed by the stable four-cell flow for $R_o < -189$. Following both of these four-cell solution branches, one stable and the other unstable, we calculated that both solution branches possess turning points. The stable branch has a turning point at $R_o \approx -99$ whereas the unstable branch has one at $R_o \approx -34$.

We also began at the three-cell Taylor vortex flow at $R_o = -85$. Increasing the outer Reynolds number, we detected a turning point at $R_o \approx -4$. This means that the three-cell flow cannot exist for $R_o = 0$, contradicting the result by Thomas indicating the three-cell flow exists $R_o = 0$. To understand this discrepancy, we duplicated the experiment on a grid which had twice the resolution. On this finer grid, we still detected a turning point, but at a slightly different outer Reynolds number, $R_o \approx 1$. The coarseness of the original grid simply caused a change in the location of the

turning point, rather than introducing an artificial turning point where one does not exist.

On the other hand, for decreasing outer Reynolds number, there is a Hopf bifurcation point at $R_o = -142.8$. The three-cell flow becomes unstable at this point, and remains unstable as R_o is decreased until $R_o = -316$, at which point it regains stability. The complex conjugate pair of eigenvalues, which moves to the right of the line $\Re e(z) = 1$ at $R_o = -142.8$, returns to the left of it at $R_o = -316$. This proves that the three-cell flow observed for $R_o < -316$ is part of the same solution branch as the three-cell flow for $R_o > -142.8$. Using only the time-stepper, we would not be able to verify this fact.

Many of these experiments were repeated using a finer 20×48 grid. Although the exact locations of the singularities moved in response to this grid refinement, the qualitative solution and bifurcation behavior described in the previous paragraphs remained unchanged.

Finally, we compared our results with those of the finite-element code ENTWIFE, a code which incorporates many bifurcation-theoretic tools. We found that the results of our code and those of ENTWIFE compared favorably for the two-cell solution branch and for flows with small outer Reynolds number. Some unexplained discrepancies still exist. We conjecture that the discrepancies may be due to the completely different computational techniques used by the two codes, the different resolutions used, the inherent difficulties in resolving fluid flows for large $|R_o|$, and the corner velocity discontinuities present in the TAY code.

7.2 Conclusions

It has been claimed that the Newton-Picard method (and the RPM) can be used as a computational wrapper around an existing time-stepper, treating the time-stepper code as a black-box about which little needs to be known [85, 121]. One of

the main goals of this dissertation was to test this claim. We conclude, in the case of the TAY time-stepper code, that this is not quite the case.

When implemented as a true black-box wrapper, early versions of NPTAY calculated spurious eigenvalues of large modulus. This caused an incorrect basis to be computed for the invariant subspace $\mathcal{P}$, and hence the entire code failed to converge. As further detailed in Appendix A, it was only through a careful understanding of the numerical method used within the TAY code that we were able to determine the cause of these spurious eigenvalues. This was possible only because we had access to the solution algorithms and source code for TAY. In general, when using a true black-box time-stepper, such intimate knowledge is either unobtainable or difficult to extract.

Once the eigenvalue calculations were corrected, the resulting program was far too slow to make a practical bifurcation code. Parts of the TAY solver routine, in particular the Poisson equation solver, were completely rewritten to increase the speed of the code. The resulting adapted program cannot be considered a true wrap-around of the TAY code. Knowledge about the time-stepper's numerical algorithms and some rewriting of the code were required to get to the final form.

In addition to modifying the original time-stepper code, we also had to make some changes to the Newton-Picard method from the way it was originally implemented. Most notable among these was the eigenvalue solver. Subspace iteration proved to be too slow and unreliable for eigenvalue computations, so the software package ARPACK was used instead. This package has been well tested and even is incorporated into the current version of MATLAB. We believe that other large-scale codes that incorporate the Newton-Picard method also would benefit from the use of this eigenvalue solver package, as it is much more efficient and robust than subspace iteration.

It is interesting to note that, although K. Lust claims the Newton-Picard method can be used as a wrapper around a black-box time-stepper, he never actually uses it in that manner. Based on our experience with the TAY code, we remain doubtful whether

the Newton-Picard method (or the related RPM) can be used as a wrapper around an arbitrary black-box time-stepper. The TAY code was written with no other application in mind other than analyzing the Taylor-Couette problem, and hence it represents a good test case for this question. To obtain a practical, working code, however, we needed to open the black-box significantly. This may be due to the nature of the TAY code itself, or it may be an indication that, contrary to the original authors' claims, these Newton-Picard methods cannot be successfully applied in general to large-scale problems which already have time simulation codes written for them.

Our results in this dissertation suggest that *some* information about the time-stepper code and/or its solution algorithm is required to form a fully functioning wrap-around code. For example, it was necessary that we knew the TAY code uses a forward explicit Euler scheme for discretizing the temporal derivative to correctly interpret the eigenvalue information for equilibrium solutions, as described in Chapter 2. Furthermore, the time-stepper program should be robust and should converge quickly to make the wrapped code practical to use. The RPM and Newton-Picard methods work well for problems where the spectrum of the linearized operator is well separated. Unfortunately, the linearization of the TAY code produces spectra where many eigenvalues are clustered near 1. This is to be expected when using a discretization for the time derivative with very small Δt. This clustering makes computing the basis of the subspace corresponding to the dangerous eigenmodes more difficult and causes the Picard iteration in the orthogonal complement still to be very slow. This latter point is especially relevant when the Newton-Picard method is combined with arclength continuation. In this case, two sets of Picard iterations must be performed for each Newton step, often at great additional cost. This clustering also contributes to the poor conditioning of the restriction of the Jacobian matrix to the large-dimensional subspace $\mathcal{Q}$, which necessitated that the Picard iteration convergence tolerance be about three orders of magnitude smaller than the Newton convergence tolerance. Unfortunately, there is no way to tell *a priori* whether a given

black-box time-stepper will produce a well-separated spectrum, unless one knows more about its computational techniques. Even with that knowledge, it seems likely that any large-scale time-stepper code will possess a linearization whose spectrum is clustered near 1 when using small Δt, based on our analysis in Chapter 2.

In the end, we see that a combination of both the original time-stepper TAY and the resulting bifurcation code NPTAY is needed to understand more fully the behavior of the Taylor-Couette problem. There are some tasks, such as following the time-dependent branch emanating from a torus bifurcation point, which only TAY can perform, and there are others which only NPTAY can do. Between the two programs, we have shed some light on a small sliver of the Taylor-Couette problem.

The forgoing discussion is not meant to suggest that these Newton-Picard methods are of little value. In fact, these methods can be quite powerful when coupled with a time-stepper code that has been written with the intent of using it within a Newton-Picard method. The work of Love on the Taylor-Couette problem and Kolmogorov flows [84] and the work of Tiesinga et al. on the driven cavity problem [137] are examples where a time-stepper was written specifically to be used in conjunction with these methods. We conclude that it remains unclear whether these methods can be applied successfully to an *arbitrary* black-box time-stepper code, one that was not written necessarily to take specific advantage of these Newton-Picard methods.

Our work in this dissertation has illuminated many of the difficulties in adapting an existing time-stepper code by using the Newton-Picard method. Others who wish to accomplish similar adaptations of time-steppers using the Newton-Picard method may benefit from our results. In particular, one should be aware of the following:

1. A small value of Δt will cause eigenvalue clustering near 1, regardless of whether an explicit or implicit method is used in the time-stepper. This clustering presents the most severe obstacle for the Newton-Picard method, for the following reasons:

(a) The clustering causes the original iteration scheme to be slowly convergent, since the rate of convergence is related to the spectral radius of the Jacobian of the iteration map. Even when projected onto the subspace $\mathcal{Q}$, the iteration still will be slowly convergent.

(b) The computation of the dangerous eigenvalues via ARPACK is more expensive when the eigenvalues are clustered. A fixed basis size for $\mathcal{P}$ will be required in this case.

(c) The $\mathcal{Q}$-equations are poorly conditioned when the eigenvalues are clustered around 1, requiring their solution to be several orders of magnitude more accurate than the convergence criterion of the Newton iteration.

2. Some information about the solution scheme used in the time-stepper must be known to correctly interpret the eigenvalue information returned by the adapted code. This requires that connections between eigenvalues of the Jacobian of the iteration map and the stability of solutions of the original dynamical system be established, as we did in Chapter 2.

3. One needs to be able to identify the independent variables/degrees of freedom so that the iteration map can be placed in the proper mathematical setting. This will typically require some information about the algorithms employed by the time-stepper. For example, we needed to know the details of the Chorin-Témam projection scheme to correctly identify the iteration map as a map from velocity space to itself, rather than from velocity-pressure space to itself.

4. The method becomes at least twice as expensive in an arclength continuation context, since two sets of equations must be solved iteratively in the large-dimensional subspace $\mathcal{Q}$. These two sets of equations have the same coefficient matrix but different right hand sides. Using direct methods that factor the coefficient matrix therefore would be more efficient than using iteration. However,

the Jacobian matrix, and its $\mathcal{Q}$-projection, are not available explicitly, so direct methods cannot be applied. Indeed, direct methods should not be used in $\mathcal{Q}$, since avoiding direct methods for these large-dimension equations is one of the objectives of using the Newton-Picard method.

5. It is an open question whether time-steppers for large dimensional problems (say $N = O(10^4)$ or $O(10^5)$) are amenable to adaptation by the Newton-Picard method, even if the eigenvalue clustering near 1 is not present.

We do not know whether there are further obstacles, other than those we have encountered already, which might arise while adapting different time-stepper codes. We can give, however, some general guidelines for selecting a time-stepper that would be well-suited for adaptation by the Newton-Picard method. In particular, the spectrum of the Jacobian of the iteration map should be well-separated. In this case, the eigenvalue solver ARPACK is more efficient, the rate of convergence in $\mathcal{Q}$ is faster, and the conditioning of the $\mathcal{Q}$-equations may be better. A time-stepper that utilizes an implicit scheme with large Δt, for instance, should yield a well-separated spectrum. The associated loss of accuracy when using a large Δt may be acceptable, especially if only equilibrium solutions are being computed. An analysis of other schemes, such as Runge-Kutta procedures, could be fruitful in identifying time-stepper codes with desirable spectral properties. Finally, for a well-suited time-stepper, it should be very cheap to evaluate one application of the iteration map, since many such iterations are necessary to compute a basis for $\mathcal{P}$, to solve the $\mathcal{Q}$-equations, and to use arclength continuation methods.

7.3 Future Work

As with any endeavor, there is much further work that can be done on the Taylor-Couette problem using our methods. First, the numerical experiments described in this dissertation should be repeated on a finer spatial grid, such as a 40×96 grid.

We emphasize that the results obtained only are results for the discrete dynamical system which approximates the continuous Taylor-Couette problem. Since both grids we used are rather coarse, the location of interesting points, in terms of Reynolds numbers, probably will change as the grid is refined. Already we have observed this when comparing results using a 10×24 grid and those using a 20×48 grid. Repeating the numerical experiments on a finer grid should help pin down the bifurcation points more accurately (relative to the continuous problems). We would also like to understand why some singularity locations change by such a large amount as the grid is refined. Recomputing their locations using even finer grids may help in our understanding of this. It may also shed some light on the discrepancies observed between the NPTAY and ENTWIFE codes.

Accomplishing these tasks using finer grids will require much more computer time and resources on top of the already considerable amount we have spent thus far. Doubling the resolution more than quadruples the size of the system being solved; for example, a 40×96 grid will have 11,115 degrees of freedom. Furthermore, decreasing the spatial resolution will typically necessitate a decrease in the time step increment Δt to maintain the stability of the forward explicit Euler scheme. Adair [2] notes that doubling the spatial resolution typically requires halving Δt. Such a small value of Δt would cause even tighter eigenvalue clustering than what we have observed already, which in turn would necessitate an increase in the number of Picard iterations required for convergence in $\mathcal{Q}$ and in the ARPACK algorithms. Thus, the computational expense would increase not only because of the increased system dimension, but also as a consequence of the tighter eigenvalue clustering. Moreover, Adair notes that the TAY code suffers from "computational difficulties" when Δt is approximately 10^{-6} [2]. This fact imposes a lower bound on the values of Δr and Δz that can be used in any spatial discretization, since decreases in the spatial discretization require reductions in the size of Δt.

Second, results should be obtained about non-axisymmetric solutions. We restricted solution computations to axisymmetric flows to speed up the code. Adair and Thomas [1, 2] used the TAY code with several Fourier θ modes to verify that the solutions found in the parameter region $R_i = 300, -320 \leq R_o \leq 0$ are axisymmetric, and hence these flows were ideal to study with our axisymmetric code. As processor speeds continue to increase, it should be feasible to extend the NPTAY code to non-axisymmetric problems. Some very interesting non-axisymmetric solutions were observed by Adair and Thomas [1, 2], and it would be worthwhile to explore them with our code. Note, however, that each additional Fourier mode included in the θ expansion adds double the storage requirement to what we used for axisymmetric solutions. This is because the values of both the cosine and sine terms at every interior gridpoint must be stored. Many of the nonaxisymmetric flows computed by Adair required at least four modes, which would mean a system dimension nine times larger than what we used in this dissertation. Presently, this would be a very slow code to run, even on a coarse grid.

Third, branch switching should be incorporated into the NPTAY code so that it is more automatic. Currently, a combination of both the TAY and NPTAY codes is required, along with manual input from the user, to switch from an unstable to a stable solution branch. This process could be more streamlined. Furthermore, the user interface of the NPTAY code is somewhat awkward, which is typical of a research code. It could be cleaned up and made more user-friendly should users other than the author desire to use the code.

We barely have scratched the surface of solution behaviors for the Taylor-Couette problem. Our code can be used to perform more numerical experiments for values of Reynolds numbers other than $R_i = 300$, $-320 \leq R_o \leq 0$. Different cylinder geometries, using various radius ratios η and aspect ratios Γ, also could be explored with the code. We have focused on counter-rotating flows in this dissertation. It would be interesting to explore co-rotating cylinders as well. Furthermore, the numerical

method used in the TAY code assumes that the top and bottom of the cylinder arrangement are stationary. As the experiments of Tavener, Mullin, and Cliffe [131] show, new bifurcation behavior can be obtained by allowing the ends to rotate with the inner cylinder. A simple modification to the TAY code would allow us to study such cylinder arrangements, and it may prove fruitful to do so where the ends rotate with the outer cylinder as well.

It would also be interesting to change the solution scheme used to solve the $\mathcal{Q}$-equations. Presently, Picard iteration is used, but many such iterations typically are needed to solve the system accurately enough. This is very expensive especially if NPTAY is run in arclength continuation mode, since *two* sets of Picard iterations are required. A more sophisticated iterative solver, such as GMRES, may prove to be more efficient than simple Picard iteration.

Finally, it would be intriguing to attempt to use the Newton-Picard method as a computational wrapper around a different large-scale time-stepper code, perhaps one produced commercially. Can this be done while still treating the time-stepper as a true black-box? We were unable to accomplish this for the TAY code; the black-box paradigm was violated many times while developing our program. Our efforts in this dissertation suggest that it is difficult to treat the time-stepper as a pure black-box, but further testing could be done to verify this. Nevertheless, our results can be used to guide future research in adapting other time-steppers, since we have overcome many, if not most, of the obstacles to that goal.

Appendix A

Spurious Eigenvalue Results

In this appendix we explain in more detail the spurious eigenvalues that were computed by early versions of the NPTAY code, as mentioned in Chapter 5. We also discuss the source of these extra eigenvalues and how this problem was resolved by viewing the time-stepper code TAY in the proper mathematical setting. The discussion in this section should serve as a warning about using time-steppers as a pure black-box when combined with a Newton-Picard method.

In early versions of our code, the solution vector $\mathbf{x}$ consisted of all velocity and pressure degrees of freedom; that is, all velocity components at all interior gridpoints plus all pressure values at all interior gridpoints. More specifically, let N denote the total number of interior gridpoints resulting from the discretization of the rz-plane. Let $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^N$ denote vectors consisting of all of the radial, azimuthal, and axial velocities evaluated at each interior gridpoint. (The particular order for arranging the gridpoints is irrelevant as long as we are consistent.) Let $\mathbf{p} \in \mathbb{R}^N$ denote the vector consisting of all pressure values evaluated at each interior gridpoint. Let $\mathbf{x} = (\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{p}) \in \mathbb{R}^{4N}$. We viewed the TAY code as being an iteration of the form

$$\mathbf{x}^{n+1} = \mathbf{F}(\mathbf{x}^n), \quad n = 0, 1, \ldots.$$

This seemed the logical mathematical setting in which to view the time-stepper.

The early versions of NPTAY utilized the finite difference approximation

$$M\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{x} + \varepsilon\mathbf{v}) - \mathbf{F}(\mathbf{x})}{\varepsilon}$$

to compute Jacobian-vector products $M\mathbf{v}$. Unfortunately, spurious eigenvalues were produced in almost every early test of the code. Hence, the basis for the invariant subspace $\mathcal{P}$ was incorrectly computed, which in turn caused the entire Newton-Picard algorithm to fail.

Consider, for example, the steady two-cell flow found at $R_i = 300, R_o = -80$, shown in figure 4.1. This solution was obtained directly from the TAY code by iterating for hundreds of thousands of time steps to guarantee that a stable solution was computed. Thus, all eigenvalues of the Jacobian matrix evaluated at this solution should have modulus less than 1. Figure A.1 shows the 15 dominant eigenvalues produced by an early version of NPTAY. There are two eigenvalues located at $-2.31 \pm 2.167i$ which clearly lie outside the unit circle. The other 13 dominant eigenvalues are clustered near 1. Since these two eigenvalues are well outside the unit circle, the TAY code could not have converged to this solution. We concluded that these extra two eigenvalues could not be physically realistic, but instead must be produced by a numerical error in the code.

Further testing revealed the following:

- The location of the spurious eigenvalues depended on the value chosen for ε, the finite difference increment used.
- The number of spurious eigenvalues produced depended on the basis convergence tolerance used within ARPACK.
- When looking at output produced directly by ARPACK, we found that every spurious eigenvalue had a residual error of exactly 0. This is curious, since we expect that any eigenvalue solver will give small but nonzero residual errors.
- There were no logical programming errors that would account for these spurious eigenvalues. This was verified later by the successful results obtained when the linearized version of TAY was written and incorporated into our code.

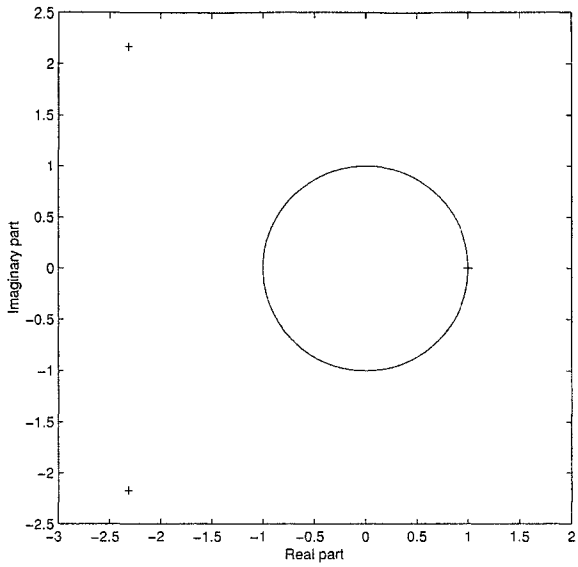

Figure A.1: *Dominant eigenvalue plot of the solution at* $R_i = 300, R_o = -80$ *showing spurious eigenvalues at* $-2.31 \pm 2.167i$. *Eigenvalues are denoted by* $+$. *The unit circle is drawn for reference.*

Even with this additional information, we could not understand the origin of the spurious eigenvalues. Thus, to salvage the project, we modified the code in two significant ways. First, we wrote a linearized version of the TAY code to implement the multiplicative action of the Jacobian matrix, rather than using finite differences (see Appendix B for details). Second, rather than just using this linearized TAY code to compute Jacobian-vector products, we explicitly constructed the entire Jacobian matrix, abandoning our original goal of a matrix-free code in the process. The Jacobian matrix was constructed by computing its multiplicative action (via the linearized TAY code) on the standard basis vectors $\mathbf{e}_i, i = 1, 2, \ldots, 4N$, where $\mathbf{e}_i$ is the vector whose only nonzero entry is a 1 in position i.

Explicitly constructing the Jacobian matrix, although costly, did produce a working code. However, the amount of time required for one computation was far too long to make a practical bifurcation code, and explicitly constructing the Jacobian matrix was later abandoned. Nevertheless, it was by explicitly constructing the Jacobian matrix that we finally understood the source of the spurious eigenvalues. We discovered

that the last N columns of the matrix were identical, and hence that the Jacobian matrix was singular. The computation of spurious eigenvalues is not an uncommon behavior when an eigenvalue solver is not applied carefully to a singular matrix, which is what occurred in our case.

The last N identical columns are obtained by computing the multiplicative action on the vectors $\mathbf{e}_i, i = 3N+1, \ldots, 4N$, indices which correspond to the pressure degrees of freedom. Regardless of which of these $\mathbf{e}_i$ are used, the multiplicative action of the Jacobian matrix on that vector is the same. By linearity, this means that the action of the Jacobian matrix is independent of any values which correspond to the pressure degrees of freedom.

Examining the Chorin-Témam projection scheme in more detail shows why this occurs. Using the same notation as in Section 4.2, the velocity $\mathbf{v}^{n+1}$ is obtained as

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t \nabla p^{n+1},$$

where

$$\mathbf{v}^* = \mathbf{v}^n + \Delta t \Big(\nabla^2 \mathbf{v}^n + L\mathbf{v}^n - R_i[(\mathbf{v}^n \cdot \nabla)\mathbf{v}^n + A(\mathbf{v}^n)] \Big).$$

Note, however, that both $\mathbf{v}^*$ and p^{n+1} are dependent on the velocity $\mathbf{v}^n$. Therefore, $\mathbf{v}^{n+1}$ depends only on $\mathbf{v}^n$; the pressure term acts as an intermediate value in the solution scheme. In fact, utilizing the formal symbol ∇^{-2} to denote the solution of the Poisson equation for p^{n+1} (with appropriate boundary conditions), we can write the projection scheme in one step as

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t \, \nabla \left(\nabla^{-2} \left(\frac{1}{\Delta t} \nabla \cdot \mathbf{v}^* \right) \right).$$

We now see why storing the pressure degrees of freedom in the solution vector $\mathbf{x}$ is unnecessary, and indeed, harmful. Contrary to our original supposition, the pressure degrees of freedom are strongly tied to the velocity degrees of freedom. Our initial view of $\mathbf{F}$ as a mapping from velocity and pressure space back to velocity and pressure space was wrong. The correct mathematical framework is to view $\mathbf{F}$ as a

mapping from velocity space to velocity space, where $\mathbf{x} = (\mathbf{u}, \mathbf{v}, \mathbf{w})$. Once this change was incorporated into the code, reducing the system dimension by one-fourth in the process, we found that using finite differences to approximate the multiplicative action of the Jacobian worked quite well, and no spurious eigenvalues were produced.

Appendix B

Details of the Linearized Solver

In this appendix we give the details of the solution technique used by the linearized Navier-Stokes solver written to compute the action of the Jacobian matrix, as discussed in Chapter 5. Recall that we can view the time-stepper code of Adair as being in the form

$$\mathbf{x}^{n+1} = \mathbf{F}(\mathbf{x}^n),$$

where $\mathbf{x}$ represents the radial, azimuthal, and axial velocities evaluated at all interior gridpoints. When evaluated at a solution $\mathbf{x}$, the Jacobian $\mathbf{F}'(\mathbf{x})$ is a linear operator acting on $\mathbb{R}^N$, so we need to compute $\mathbf{F}'(\mathbf{x})\mathbf{u}$ for arbitrary $\mathbf{u} \in \mathbb{R}^N$.

To do this, we linearize the Navier-Stokes equations about the solution $\mathbf{x}$ and apply the same solution technique (the Chorin-Témam projection scheme) for these linearized equations. We begin by writing the vector $\mathbf{u}$ in a partitioned form $\mathbf{u} = (\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}})$ and $\mathbf{x}$ as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_x)$. Here each of $\tilde{\mathbf{u}}, \tilde{\mathbf{v}}$, and $\tilde{\mathbf{w}}$ has dimension $N/3$, so that the vector $\mathbf{u}$ is partitioned in the same way as the solution vector $\mathbf{x}$. This partition of $\mathbf{x}$ corresponds to the three independent velocity directions in cylindrical coordinates. Using similar notation to that in Section 4.2, we can write $\mathbf{F}'(\mathbf{x})\mathbf{u} = \mathbf{a}$, where

$$\mathbf{a} = \mathbf{u} + \Delta t\Big[\nabla^2\mathbf{u} + L\mathbf{u} - R_i[(\mathbf{x}\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{x} + A'(\mathbf{x})\mathbf{u}]\Big] - \Delta t\nabla\mathfrak{P}, \qquad \text{(B.1)}$$

$\mathfrak{P}$ is the solution to

$$\nabla^2\mathfrak{P} = \frac{1}{\Delta t}\nabla\cdot\left\{\mathbf{u} + \Delta t\Big[\nabla^2\mathbf{u} + L\mathbf{u} - R_i[(\mathbf{x}\cdot\nabla)\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{x} + A'(\mathbf{x})\mathbf{u}]\Big]\right\} \qquad \text{(B.2)}$$

(with the same boundary conditions), the linear operator L and the cylindrical operators ∇ and ∇^2 are as defined in Section 4.2, and the action of the derivative $A'(\mathbf{x})$ is defined by

$$\begin{aligned} A'(\mathbf{x})\mathbf{u} &= A'(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}}) \\ &= \left(\frac{-2\mathbf{x}_2\tilde{\mathbf{v}}}{r}, \frac{\mathbf{x}_1\tilde{\mathbf{v}} + \mathbf{x}_2\tilde{\mathbf{u}}}{r}, \mathbf{0} \right). \end{aligned}$$

Comparing this with the original scheme used by Adair, we see that the two are almost the same, except that to solve for the "pressure" $\mathfrak{P}$, we must deal with two terms of the form $(* \cdot \nabla)*$ rather than just one. Hence, the modifications necessary to carry this solution scheme out were not significant.

Recall that we are considering only axisymmetric flows; in particular, all derivatives involving θ are zero, and the truncated Fourier series given in Section 4.2 have only the constant term. To assure maximum compatibility and accuracy, these linearized equations were discretized in exactly the same way as Adair did. Centered differences are used for all second derivatives, and first derivatives are handled using a Lax-Wendroff-like scheme [135]. Terms of the form $a\frac{\partial u}{\partial r}$ evaluated at the j, k gridpoint are discretized as follows:

$$a\left(\frac{\partial u}{\partial r}\right)_{j,k} \approx a\frac{u_{j+1,k} - u_{j-1,k}}{2\Delta r} - \frac{1}{2}a^2\frac{u_{j+1,k} - 2u_{j,k} + u_{j-1,k}}{\Delta r^2}.$$

Here j refers to the gridpoint in the radial direction, $1 \leq j \leq R-1$, and k refers to the gridpoint in the axial direction, $1 \leq k \leq Z-1$. A similar discretization was used for derivatives with respect to z.

Let $\mathbf{a}^* = \mathbf{a} + \Delta t \nabla \mathfrak{P}$; that is, $\mathbf{a}^*$ is the right-hand side of (B.1) without the $\Delta t \nabla \mathfrak{P}$ term. Write $\mathbf{a}^* = (\mathbf{a}_1^*, \mathbf{a}_2^*, \mathbf{a}_3^*)$, partitioned in the same way as $\mathbf{x}$ and $\mathbf{u}$. At the j, k gridpoint we get the following formulas for the constant Fourier modes of $\mathbf{a}^*$:

$$
\begin{aligned}
(\mathbf{a}_1^*)_{j,k} =& \tilde{\mathbf{u}}_{j,k} + \frac{\Delta t}{r_j}\frac{\tilde{\mathbf{u}}_{j+1,k}-\tilde{\mathbf{u}}_{j-1,k}}{2\Delta r} - \frac{1}{2}\left(\frac{\Delta t}{r_j}\right)^2 \frac{\tilde{\mathbf{u}}_{j+1,k}-2\tilde{\mathbf{u}}_{j,k}+\tilde{\mathbf{u}}_{j-1,k}}{\Delta r^2} \\
&+ \Delta t\frac{\tilde{\mathbf{u}}_{j+1,k}-2\tilde{\mathbf{u}}_{j,k}+\tilde{\mathbf{u}}_{j-1,k}}{\Delta r^2} + \Delta t\frac{\tilde{\mathbf{u}}_{j,k+1}-2\tilde{\mathbf{u}}_{j,k}+\tilde{\mathbf{u}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t}{r_j^2}\tilde{\mathbf{u}}_{j,k} + \frac{\Delta t R_i}{r_j}(\mathbf{x}_2)_{j,k}\tilde{\mathbf{v}}_{j,k} - \frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\frac{\tilde{\mathbf{u}}_{j+1,k}-\tilde{\mathbf{u}}_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\right)^2 \frac{\tilde{\mathbf{u}}_{j+1,k}-2\tilde{\mathbf{u}}_{j,k}+\tilde{\mathbf{u}}_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\frac{\tilde{\mathbf{u}}_{j,k+1}-\tilde{\mathbf{u}}_{j,k-1}}{2\Delta z} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\right)^2 \frac{\tilde{\mathbf{u}}_{j,k+1}-2\tilde{\mathbf{u}}_{j,k}+\tilde{\mathbf{u}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\frac{(\mathbf{x}_1)_{j+1,k}-(\mathbf{x}_1)_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\right)^2 \frac{(\mathbf{x}_1)_{j+1,k}-2(\mathbf{x}_1)_{j,k}+(\mathbf{x}_1)_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\frac{(\mathbf{x}_1)_{j,k+1}-(\mathbf{x}_1)_{j,k-1}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\right)^2 \frac{(\mathbf{x}_1)_{j,k+1}-2(\mathbf{x}_1)_{j,k}+(\mathbf{x}_1)_{j,k-1}}{\Delta z^2},
\end{aligned}
\tag{B.3}
$$

$$
\begin{aligned}
(\mathbf{a}_2^*)_{j,k} =& \tilde{\mathbf{v}}_{j,k} + \frac{\Delta t}{r_j}\frac{\tilde{\mathbf{v}}_{j+1,k}-\tilde{\mathbf{v}}_{j-1,k}}{2\Delta r} - \frac{1}{2}\left(\frac{\Delta t}{r_j}\right)^2 \frac{\tilde{\mathbf{v}}_{j+1,k}-2\tilde{\mathbf{v}}_{j,k}+\tilde{\mathbf{v}}_{j-1,k}}{\Delta r^2} \\
&+ \Delta t\frac{\tilde{\mathbf{v}}_{j+1,k}-2\tilde{\mathbf{v}}_{j,k}+\tilde{\mathbf{v}}_{j-1,k}}{\Delta r^2} + \Delta t\frac{\tilde{\mathbf{v}}_{j,k+1}-2\tilde{\mathbf{v}}_{j,k}+\tilde{\mathbf{v}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t}{r_j^2}\tilde{\mathbf{v}}_{j,k} - \frac{\Delta t R_i}{2r_j}(\mathbf{x}_1)_{j,k}\tilde{\mathbf{v}}_{j,k} - \frac{\Delta t R_i}{2r_j}(\mathbf{x}_2)_{j,k}\tilde{\mathbf{u}}_{j,k} \\
&- \frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\frac{\tilde{\mathbf{v}}_{j+1,k}-\tilde{\mathbf{v}}_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\right)^2 \frac{\tilde{\mathbf{v}}_{j+1,k}-2\tilde{\mathbf{v}}_{j,k}+\tilde{\mathbf{v}}_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\frac{\tilde{\mathbf{v}}_{j,k+1}-\tilde{\mathbf{v}}_{j,k-1}}{2\Delta z} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\right)^2 \frac{\tilde{\mathbf{v}}_{j,k+1}-2\tilde{\mathbf{v}}_{j,k}+\tilde{\mathbf{v}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\frac{(\mathbf{x}_2)_{j+1,k}-(\mathbf{x}_2)_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\right)^2 \frac{(\mathbf{x}_2)_{j+1,k}-2(\mathbf{x}_2)_{j,k}+(\mathbf{x}_2)_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\frac{(\mathbf{x}_2)_{j,k+1}-(\mathbf{x}_2)_{j,k-1}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\right)^2 \frac{(\mathbf{x}_2)_{j,k+1}-2(\mathbf{x}_2)_{j,k}+(\mathbf{x}_2)_{j,k-1}}{\Delta z^2},
\end{aligned}
\tag{B.4}
$$

$$
\begin{aligned}
(\mathbf{a}_3^*)_{j,k} =& \tilde{\mathbf{w}}_{j,k} + \frac{\Delta t}{r_j}\frac{\tilde{\mathbf{w}}_{j+1,k} - \tilde{\mathbf{w}}_{j-1,k}}{2\Delta r} - \frac{1}{2}\left(\frac{\Delta t}{r_j}\right)^2 \frac{\tilde{\mathbf{w}}_{j+1,k} - 2\tilde{\mathbf{w}}_{j,k} + \tilde{\mathbf{w}}_{j-1,k}}{\Delta r^2} \\
&+ \Delta t \frac{\tilde{\mathbf{w}}_{j+1,k} - 2\tilde{\mathbf{w}}_{j,k} + \tilde{\mathbf{w}}_{j-1,k}}{\Delta r^2} + \Delta t \frac{\tilde{\mathbf{w}}_{j,k+1} - 2\tilde{\mathbf{w}}_{j,k} + \tilde{\mathbf{w}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\frac{\tilde{\mathbf{w}}_{j+1,k} - \tilde{\mathbf{w}}_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_1)_{j,k}\right)^2 \frac{\tilde{\mathbf{w}}_{j+1,k} - 2\tilde{\mathbf{w}}_{j,k} + \tilde{\mathbf{w}}_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\frac{\tilde{\mathbf{w}}_{j,k+1} - \tilde{\mathbf{w}}_{j,k-1}}{2\Delta z} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}(\mathbf{x}_3)_{j,k}\right)^2 \frac{\tilde{\mathbf{w}}_{j,k+1} - 2\tilde{\mathbf{w}}_{j,k} + \tilde{\mathbf{w}}_{j,k-1}}{\Delta z^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\frac{(\mathbf{x}_3)_{j+1,k} - (\mathbf{x}_3)_{j-1,k}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{u}}_{j,k}\right)^2 \frac{(\mathbf{x}_3)_{j+1,k} - 2(\mathbf{x}_3)_{j,k} + (\mathbf{x}_3)_{j-1,k}}{\Delta r^2} \\
&- \frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\frac{(\mathbf{x}_3)_{j,k+1} - (\mathbf{x}_3)_{j,k-1}}{2\Delta r} \\
&+ \frac{1}{2}\left(\frac{\Delta t R_i}{2}\tilde{\mathbf{w}}_{j,k}\right)^2 \frac{(\mathbf{x}_3)_{j,k+1} - 2(\mathbf{x}_3)_{j,k} + (\mathbf{x}_3)_{j,k-1}}{\Delta z^2}.
\end{aligned}
\tag{B.5}
$$

These equations (B.3), (B.4), and (B.5) determine $\mathbf{a}^*$. The modified Poisson solver in TAY is also used to solve (B.2) for $\mathfrak{P}$. Once $\mathbf{a}^*$ and $\mathfrak{P}$ are computed, we compute the discrete gradient of $\mathfrak{P}$ and combine them to obtain

$$\mathbf{a} = \mathbf{F}'(\mathbf{x})\mathbf{u} = \mathbf{a}^* - \Delta t \nabla \mathfrak{P}.$$

Appendix C

Fast Solution of the Poisson Equation

In this appendix, we discuss the changes made to the TAY program to speed it up, and in particular, we detail the new solver routine for the Poisson equations (4.5) and (B.2).

Let us recap the details of the Chorin-Témam projection scheme. In cylindrical coordinates, the vector form of the Navier-Stokes equations can be written as

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} &= \nabla^2 \mathbf{v} + L\mathbf{v} - \nabla p - R_i[(\mathbf{v}\cdot\nabla)\mathrm{v} + A(\mathbf{v})], \\ \nabla \cdot \mathbf{v} &= 0, \end{aligned} \tag{C.1}$$

where the operators appearing on the right-hand side of (C.1) are in cylindrical coordinates and are defined in Chapter 4. The Chorin-Témam scheme is the following three-step scheme. Upon discretization of the temporal derivative $\frac{d\mathrm{v}}{dt}$ by the forward Euler scheme, we let

$$\mathbf{v}^* = \mathbf{v}^n + \Delta t\Big(\nabla^2\mathbf{v}^n + L\mathbf{v}^n - R_i[(\mathbf{v}^n\cdot\nabla)\mathbf{v}^n + A(\mathbf{v}^n)]\Big).$$

The Poisson equation for the pressure is

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t}\nabla\cdot\mathbf{v}^*, \tag{C.2}$$

which is solved for p^{n+1} using boundary conditions

$$\left(\frac{dp}{d\mathbf{n}}\right)^{n+1} = -\frac{1}{\Delta t}\,\mathbf{n}\cdot(\mathbf{v}^{n+1} - \mathbf{v}^*), \tag{C.3}$$

where $\mathbf{n}$ is the outward pointing normal vector. Finally, the velocity at the next time step is computed as

$$\mathbf{v}^{n+1} = \mathbf{v}^* - \Delta t \nabla p^{n+1}. \tag{C.4}$$

We will focus our attention on the solution of (C.2) with boundary conditions (C.3). The Gauss-Seidel scheme, an iterative method for solving linear equations, is used in TAY to solve this Poisson equation. Note that this is a problem with Neumann boundary conditions. Because of the normal derivative of the pressure appearing in (C.3), the solution is not unique; it is determined only up to a constant. The TAY code deals with this non-uniqueness by normalizing the pressure so that its average value over the entire rz-plane, including boundaries, is zero.

As we explained in Chapter 5, using the Gauss-Seidel method to solve the Poisson equation proved to be unacceptably slow for use in the NPTAY wrap-around code. Our solution was to rewrite the solver and use a direct method to solve the equation rather than an iterative one, the details of which we now explain.

C.1 The Matrix Equation

Let us first rewrite equation (C.2) in the equivalent form

$$-\Delta t \nabla^2 p^{n+1} = -\nabla \cdot \mathbf{v}^*. \tag{C.5}$$

This is the form of the Poisson equation that is actually solved by TAY. Consider a grid on the rz-plane which has R total nodes in the radial direction and Z total nodes in the axial direction (including all boundaries), and let Δr and Δz denote the partition width in the r and z directions, respectively. Let p_{jk} denote the pressure value at gridpoint $(j\Delta r, k\Delta z)$, for $0 \leq j \leq R$, $0 \leq k \leq Z$. Let

$$\mathbf{u} = (p_{00}, p_{01}, \ldots, p_{RZ})^T$$

denote the vector of dimension $L = RZ$ containing the pressure values at every gridpoint. Finally, let $\mathbf{f}$ denote the vector in $\mathbb{R}^L$ which is obtained by applying a

second order finite difference approximation to $-\nabla \cdot \mathbf{v}^*$. Using exactly the same stencils as the TAY code, upon this discretization equation (C.5) becomes the matrix equation

$$A\mathbf{u} = \mathbf{f}. \tag{C.6}$$

The $L \times L$ matrix A in C.6 is called the matrix of coefficients for the discretized Poisson equation (C.5). Figure C.1 gives the general (block) form of the matrix of coefficients. Each block row has R blocks, each of which has size $Z \times Z$. The constants which appear in the matrix are defined as

$$\alpha_r = \frac{\Delta t}{4\Delta r^2}, \quad \alpha_z = \frac{\Delta t}{4\Delta z^2}, \quad \text{and} \quad \beta_j = \frac{\Delta t}{2r_j \Delta r},$$

and the blocks themselves are defined by:

$$A_1 = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & \cdots & 0 \\ 0 & 4\alpha_r & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & 4\alpha_r & 0 \\ 0 & \cdots & 0 & -\frac{1}{2} & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} -\frac{1}{2} & & & & \\ & -\alpha_r & & & \\ & & \ddots & & \\ & & & -\alpha_r & \\ & & & & -\frac{1}{2} \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 4\alpha_z & -\alpha_z & -4\alpha_z & \alpha_z & 0 & \cdots & 0 \\ 0 & \alpha_r + \alpha_z & 0 & -\alpha_z & 0 & \cdots & 0 \\ -\alpha_z & 0 & \alpha_r + 2\alpha_z & 0 & -\alpha_z & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -\alpha_z & 0 & \alpha_r + 2\alpha_z & 0 & -\alpha_z \\ 0 & \cdots & 0 & -\alpha_z & 0 & \alpha_r + \alpha_z & 0 \\ 0 & \cdots & 0 & \alpha_z & -4\alpha_z & -\alpha_z & 4\alpha_z \end{bmatrix},$$

$$A_4 = \begin{bmatrix} 0 & & & & & & \\ & -\alpha_r & & & & & \\ & & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & & \\ & & & & & -\alpha_r & \\ & & & & & & 0 \end{bmatrix},$$

$$A_5 = \begin{bmatrix} 4\alpha_z & -\alpha_z & -4\alpha_z & \alpha_z & 0 & \cdots & 0 \\ 0 & 2\alpha_r + \alpha_z & 0 & -\alpha_z & 0 & \cdots & 0 \\ 0 & -4\alpha_z & 8\alpha_r + 8\alpha_z & -4\alpha_z & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & -4\alpha_z & 8\alpha_r + 8\alpha_z & -4\alpha_z & 0 \\ 0 & \cdots & 0 & -\alpha_z & 0 & 2\alpha_r + \alpha_z & 0 \\ 0 & \cdots & 0 & \alpha_z & -4\alpha_z & -\alpha_z & 4\alpha_z \end{bmatrix},$$

$$B(j) = \begin{bmatrix} \beta_j & & & & \\ & \beta_j - 4\alpha_r & & & \\ & & \ddots & & \\ & & & \beta_j - 4\alpha_r & \\ & & & & \beta_j \end{bmatrix},$$

and

$$C(j) = \begin{bmatrix} -\beta_j & & & & \\ & -\beta_j - 4\alpha_r & & & \\ & & \ddots & & \\ & & & -\beta_j - 4\alpha_r & \\ & & & & -\beta_j \end{bmatrix}.$$

The form of $\mathbf{f}$ depends on whether it is equation (C.5) or the analog of (B.2) that we are solving. In either case, the matrix A is the same.

C.1.1 Properties of the Matrix of Coefficients

The matrix of coefficients A possesses several important properties, which we summarize in the following proposition.

Proposition C.1. *Let A denote the $L \times L$ matrix of coefficients of the discrete Poisson equation, as given above. Then:*

(a) the rank of A is $L - 1$;

(b) $\mathcal{N}(A) = \text{span}\{\mathbf{1}\}$, where $\mathbf{1} = (1, \ldots, 1)^T \in \mathbb{R}^L$;

$$
\begin{bmatrix}
A_1 & A_2 & \begin{matrix}0 & & \\ & -4\alpha_r I & \\ & & 0\end{matrix} & \begin{matrix}0 & & \\ & \alpha_r I & \\ & & 0\end{matrix} & 0 & 0 & 0 & \cdots & 0 \\
\begin{matrix}0 & & \\ & \beta_1 I & \\ & & 0\end{matrix} & A_3 & \begin{matrix}0 & & \\ & -\beta_1 I & \\ & & 0\end{matrix} & \begin{matrix}0 & & \\ & -\alpha_r I & \\ & & 0\end{matrix} & 0 & 0 & 0 & \cdots & 0 \\
A_4 & \begin{matrix}0 & & \\ & B(2) & \\ & & 0\end{matrix} & A_5 & \begin{matrix}0 & & \\ & C(2) & \\ & & 0\end{matrix} & A_4 & 0 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & & & \vdots \\
0 & \cdots & A_4 & \begin{matrix}0 & & \\ & B(j) & \\ & & 0\end{matrix} & A_5 & \begin{matrix}0 & & \\ & C(j) & \\ & & 0\end{matrix} & A_4 & \cdots & 0 \\
\vdots & & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 0 & A_4 & \begin{matrix}0 & & \\ & B(R-3) & \\ & & 0\end{matrix} & A_5 & \begin{matrix}0 & & \\ & C(R-3) & \\ & & 0\end{matrix} & A_4 \\
0 & \cdots & 0 & 0 & 0 & \begin{matrix}0 & & \\ & -\alpha_r I & \\ & & 0\end{matrix} & \begin{matrix}0 & & \\ & \beta_{R-2} I & \\ & & 0\end{matrix} & A_3 & \begin{matrix}0 & & \\ & -\beta_{R-2} I & \\ & & 0\end{matrix} \\
0 & \cdots & 0 & 0 & 0 & \begin{matrix}0 & & \\ & \alpha_r I & \\ & & 0\end{matrix} & \begin{matrix}0 & & \\ & -4\alpha_r I & \\ & & 0\end{matrix} & A_2 & A_1
\end{bmatrix}
$$

Figure C.1: *The matrix of coefficients for the discrete Laplacian. Each of the R blocks in each row is $Z \times Z$.*

(c) the linear equation $A\mathbf{u} = \mathbf{f}$ *has a solution if and only if* $\langle \mathbf{f}, \mathbf{u}^* \rangle = 0$, *where* $\mathbf{u}^*$ *is a nonzero vector such that* $A^T\mathbf{u}^* = \mathbf{0}$, *and* $\langle \, , \rangle$ *denotes the standard inner product on* $\mathbb{R}^L$.

PROOF: (a) The same proof as in [136], page 375, can be applied to this matrix.

(b) By part (a), we know that $\dim \mathcal{N}(A) = 1$. A simple computation verifies that $A\mathbf{1} = \mathbf{0}$, from which we conclude statement (b).

(c) This is a restatement of part of the Fredholm Alternative for finite dimensional spaces, which follows from the fact that the range of A is the orthogonal complement of the null space of A^T. Note that $\dim \mathcal{N}(A^T) = 1$ also, so if $\mathbf{u}^*$ is any nonzero vector in $\mathcal{N}(A^T)$, then $\mathcal{N}(A^T) = \text{span}\{\mathbf{u}^*\}$. ■

Part (b) of Proposition C.1 implies that, when it exists, a solution of $A\mathbf{u} = \mathbf{f}$ is not unique. This is the linear algebra analog of the non-uniqueness of the solution to the original PDE Neumann problem. How, then, can a solution to $A\mathbf{u} = \mathbf{f}$ be obtained? The key lies in the next result.

Proposition C.2 (from Thomas [136], page 382). *Consider the augmented linear system* $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$, *where*

$$\bar{A} = \begin{bmatrix} A & \mathbf{u}^* \\ \mathbf{1}^T & 0 \end{bmatrix}, \quad \bar{\mathbf{u}} = \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix}, \quad \text{and} \quad \bar{\mathbf{f}} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix},$$

and where $\mathbf{0} \neq \mathbf{u}^* \in \mathcal{N}(A^T)$ *as in Proposition C.1. Then:*

(a) The system $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$ *has a unique solution.*

(b) If the solution to the system $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$ *is of the form* $\bar{\mathbf{u}}_0 = (\mathbf{u}_0, \lambda)^T$, *where* $\lambda \neq 0$, *then* $\mathbf{u}_0$ *is the unique solution of the equation*

$$A\mathbf{u} = \mathbf{f} - \lambda\mathbf{u}^* \tag{C.7}$$

such that $\langle \mathbf{u}_0, \mathbf{1} \rangle = 0$.

PROOF: (a) Because $\mathcal{N}(A^T) = \mathcal{R}(A)^\perp$, we have that $\mathbf{u}^* \in \mathcal{R}(A)^\perp$. Since the range of A is the span of the columns of A, this implies that $\mathbf{u}^*$ is independent of the columns of A, and hence the rank of $\begin{bmatrix} A & \mathbf{u}^* \end{bmatrix}$ is one more than the rank of A. Since $\text{rank}(A) = L - 1$ from Proposition C.1, we get that rank $\begin{bmatrix} A & \mathbf{u}^* \end{bmatrix} = L$. Similarly, since $\mathcal{N}(A) = \mathcal{R}(A^T)^\perp$, we know that $\mathbf{1} \in \mathcal{R}(A^T)^\perp$ and hence $\mathbf{1}$ is independent of the columns of A^T; equivalently, $\mathbf{1}^T$ is independent of the rows of A. Therefore, $\begin{bmatrix} \mathbf{1}^T & 0 \end{bmatrix}$ is independent of the rows of $\begin{bmatrix} A & \mathbf{u}^* \end{bmatrix}$, and therefore, the rank of $\bar{A}$ is $L+1$. Since $\bar{A}$ is an $(L+1) \times (L+1)$ matrix, this implies that it has full rank, and hence the system $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$ is uniquely solvable.

(b) This follows easily by looking at the first L rows of the system $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$ and the last row of the system (in block matrix form). ■

C.1.2 Two Methods to Solve $A\mathbf{u} = \mathbf{f}$

In our situation, there are two difficulties in trying to solve the linear system $A\mathbf{u} = \mathbf{f}$. The first is that this system does not have a solution unless $\langle f, \mathbf{u}^* \rangle = \mathbf{0}$, which generally does not occur. The other difficulty is the fact that the solution, if it exists, is not unique. Thomas [136] gives two suggestions to deal with these difficulties.

Suppose that $\langle f, \mathbf{u}^* \rangle \neq \mathbf{0}$, that is, $A\mathbf{u} = \mathbf{f}$ does not have a solution. Proposition C.2 gives us an idea of how to proceed. Suppose that the unique solution to $\bar{A}\bar{\mathbf{u}} = \bar{\mathbf{f}}$ is of the form $\bar{\mathbf{u}}_0 = (\mathbf{u}_0, \lambda)^T$ with $\lambda \neq 0$. Then,

$$A\mathbf{u}_0 = \mathbf{f} - \lambda \mathbf{u}^*. \tag{C.8}$$

Since equation (C.8) does have a solution, it follows that $\langle \mathbf{f} - \lambda \mathbf{u}^*, \mathbf{u}^* \rangle = 0$. This allows us to solve for λ to get

$$\lambda = \frac{1}{\langle \mathbf{u}^*, \mathbf{u}^* \rangle} \langle \mathbf{f}, \mathbf{u}^* \rangle.$$

Thus, if we replace $\mathbf{f}$ by

$$\tilde{\mathbf{f}} = \mathbf{f} - \frac{\langle \mathbf{f}, \mathbf{u}^* \rangle}{\langle \mathbf{u}^*, \mathbf{u}^* \rangle} \mathbf{u}^*, \tag{C.9}$$

then the system $A\mathbf{u} = \tilde{\mathbf{f}}$ *is* solvable, and the resulting solution still will be an approximation to the continuous solution of the original PDE (C.2). Proposition C.2 and the preceding discussion implies that if we adjust the right-hand side of the equation from $\mathbf{f}$ to $\tilde{\mathbf{f}}$, we can apply any techniques we desire to solve the system $A\mathbf{u} = \tilde{\mathbf{f}}$, which does have a solution. As we see in the next section, this amounts to solving the original equation $A\mathbf{u} = \mathbf{f}$ in a least squares sense.

However, Adair computed neither the matrix of coefficients A nor the vector $\mathbf{u}^*$, so the adjustment to the right-hand side suggested above was not made. Instead, the TAY code adjusts the solution after each iteration of the Gauss-Seidel method, which is the second method described by Thomas. Suppose $\mathbf{u}$ is a solution to $A\mathbf{u} = \mathbf{f}$. Let $\mathbf{u}_0 = \mathbf{u} - \lambda^* \mathbf{1}$, where $\lambda^* = \frac{\langle \mathbf{u}, \mathbf{1} \rangle}{\langle \mathbf{1}, \mathbf{1} \rangle}$. Then,

$$A\mathbf{u}_0 = A\mathbf{u} - \lambda^* A\mathbf{1} = A\mathbf{u} = \mathbf{f},$$

since $A\mathbf{1} = \mathbf{0}$. This suggests how to adjust the solution at each step. At the end of every Gauss-Seidel iteration, project the solution $\mathbf{u}$ obtained at that step onto $\mathbf{1}$ and subtract off this projection. The resulting vector will be the solution $\mathbf{u}_0$ to $A\mathbf{u} = \mathbf{f}$ with the property that $\langle \mathbf{u}_0, \mathbf{1} \rangle = 0$. This is precisely what the TAY code does.

C.2 A Least Squares Approach

The approach we used to rewrite TAY's Poisson solver algorithm was to solve $A\mathbf{u} = \mathbf{f}$ in a least squares sense, which computes a vector $\tilde{\mathbf{u}}$ such that $\|\mathbf{f} - A\tilde{\mathbf{u}}\|$ is a minimum. Unfortunately, because A is rank deficient by one, such a least squares solution is not unique—any vector of the form $\tilde{\mathbf{u}} + c\mathbf{1}$ also minimizes the above norm and is therefore a least squares solution.

C.2.1 The Pseudo-Inverse

Using the pseudo-inverse of A allows us to find some sort of unique least squares solution to $A\mathbf{u} = \mathbf{f}$. The pseudo-inverse, sometimes called the Moore-Penrose inverse, easily can be obtained via the singular value decomposition of A [55]. The pseudo-inverse of A, denoted by A^+, enjoys many nice properties, the most important of which (for us) is that the vector $\tilde{\mathbf{u}} = A^+\mathbf{f}$ is the unique least squares solution to $A\mathbf{u} = \mathbf{f}$ which has minimum Euclidean norm. By minimum norm, we mean that if $\tilde{\mathbf{u}}'$ is another (distinct) least squares solution to $A\mathbf{u} = \mathbf{f}$, then $\|\tilde{\mathbf{u}}\| < \|\tilde{\mathbf{u}}'\|$.

The vector $\tilde{\mathbf{u}} = A^+\mathbf{f}$ plays an important role in our situation, which we summarize in the following proposition.

Proposition C.3. *Using the cumulative notation above, let* $\mathbf{u}_0$ *denote the unique solution to* $A\mathbf{u} = \tilde{\mathbf{f}}$ *with the property* $\langle \mathbf{u}_0, \mathbf{1} \rangle = 0$*, as guaranteed by Proposition C.2, where* $\tilde{\mathbf{f}}$ *is defined in equation* (C.9)*. Let* $\tilde{\mathbf{u}} = A^+\mathbf{f}$*. Then,* $\mathbf{u}_0 = \tilde{\mathbf{u}}$.

PROOF: First, let $\mathbf{u}' = A^+\tilde{\mathbf{f}}$. Then $\mathbf{u}'$ is the unique least squares solution to $A\mathbf{u} = \tilde{\mathbf{f}}$ with minimum norm. Since $A\mathbf{u}_0 = \tilde{\mathbf{f}}$, it follows that $\mathbf{u}_0$ is also a least squares solution to $A\mathbf{u} = \tilde{\mathbf{f}}$. Since $\mathcal{N}(A) = \text{span}\{\mathbf{1}\}$, we have that $\mathbf{u}' = \mathbf{u}_0 + c\mathbf{1}$ for some constant c. We also know that $\|\mathbf{u}'\| \leq \|\mathbf{u}_0\|$. This implies that $\|\mathbf{u}_0 + c\mathbf{1}\| \leq \|\mathbf{u}_0\|$ or $\|\mathbf{u}_0 + c\mathbf{1}\|^2 \leq \|\mathbf{u}_0\|^2$. Expressed in terms of inner products, this last inequality becomes

$$\langle \mathbf{u}_0 + c\mathbf{1}, \mathbf{u}_0 + c\mathbf{1} \rangle \leq \langle \mathbf{u}_0, \mathbf{u}_0 \rangle$$

or

$$\langle \mathbf{u}_0, \mathbf{u}_0 \rangle + 2\langle \mathbf{u}_0, c\mathbf{1} \rangle + c^2\langle \mathbf{1}, \mathbf{1} \rangle \leq \langle \mathbf{u}_0, \mathbf{u}_0 \rangle,$$

and since $\langle \mathbf{u}_0, \mathbf{1} \rangle = 0$, we have

$$c^2\langle \mathbf{1}, \mathbf{1} \rangle \leq 0.$$

Since $\langle \mathbf{1}, \mathbf{1} \rangle = L > 0$, we conclude that $c^2 \leq 0$ and hence $c = 0$. Therefore, $\mathbf{u}' = \mathbf{u}_0$, so $\mathbf{u}_0 = A^+ \tilde{\mathbf{f}}$.

Next, let $\lambda^* = \frac{\langle \mathbf{f}, \mathbf{u}^* \rangle}{\langle \mathbf{u}^*, \mathbf{u}^* \rangle}$, so $\tilde{\mathbf{f}} = \mathbf{f} - \lambda^* \mathbf{u}^*$. Then,

$$\mathbf{u}_0 = A^+ \tilde{\mathbf{f}} = A^+ \mathbf{f} - \lambda^* A^+ \mathbf{u}^* = \tilde{\mathbf{u}} - \lambda^* A^+ \mathbf{u}^*,$$

and hence $\tilde{\mathbf{u}} - \mathbf{u}_0 = \lambda^* A^+ \mathbf{u}^*$. Let $\mathbf{x}^* = A^+ \mathbf{u}^*$. We claim that $\mathbf{x}^* = \mathbf{0}$. Indeed, by the properties of the pseudo-inverse, we know that $\mathbf{x}^*$ is the unique least squares solution to $A\mathbf{x} = \mathbf{u}^*$ of minimum norm, which means that $A\mathbf{x}^* - \mathbf{u}^*$ is orthogonal to the range of A. Furthermore, since $\mathbf{u}^* \in \mathcal{N}(A) = \mathcal{R}(A)^\perp$, we have that $\mathbf{u}^*$ is also orthogonal to the range of A. These two facts imply that for every $\mathbf{x} \in \mathbb{R}^L$,

$$\langle \mathbf{u}^*, A\mathbf{x} \rangle = 0$$

and

$$\langle A\mathbf{x}^* - \mathbf{u}^*, A\mathbf{x} \rangle = 0.$$

From this we conclude that for every $\mathbf{x} \in \mathbb{R}^L$,

$$\begin{aligned} 0 &= \langle A\mathbf{x}^* - \mathbf{u}, A\mathbf{x} \rangle \\ &= \langle A\mathbf{x}^*, A\mathbf{x} \rangle - \langle \mathbf{u}^*, A\mathbf{x} \rangle \\ &= \langle A\mathbf{x}^*, A\mathbf{x} \rangle. \end{aligned}$$

In particular, this is true for $\mathbf{x} = \mathbf{x}^*$, which implies that $\langle A\mathbf{x}^*, A\mathbf{x}^* \rangle = 0$, so $A\mathbf{x}^* = \mathbf{0}$. This means that $\mathbf{x}^* \in \mathcal{N}(A) = \text{span}\{\mathbf{1}\}$ and so $\mathbf{x}^* = k\mathbf{1}$ for some constant k. Since any other least squares solution to $A\mathbf{x} = \mathbf{u}^*$ differs from $\mathbf{x}^*$ by a multiple of $\mathbf{1}$ and since $\mathbf{x}^*$ has minimum norm, it follows that k must be zero and that $\mathbf{x}^* = \mathbf{0}$. Thus,

$$\tilde{\mathbf{u}} - \mathbf{u}_0 = \lambda^* A^+ \mathbf{u}^* = \lambda^* \mathbf{x}^* = \mathbf{0},$$

which completes the proof. ■

In conclusion, we see that using a least squares approach for $A\mathbf{u} = \mathbf{f}$ based on the SVD of A is equivalent to finding the unique solution of $A\mathbf{u} = \tilde{\mathbf{f}} = \mathbf{f} - \frac{\langle \mathbf{f}, \mathbf{u}^* \rangle}{\langle \mathbf{u}^*, \mathbf{u}^* \rangle} \mathbf{u}^*$ which has the property $\langle \mathbf{u}_0, \mathbf{1} \rangle = \mathbf{0}$, the latter being Thomas' suggested method for solving the discrete Poisson equation with Neumann boundary conditions [136].

C.2.2 Implementation in NPTAY

The implementation of the pseudo-inverse within the NPTAY code was straightforward. MATLAB was used to generate the matrix of coefficients A and its pseudo-inverse A^+, given a particular cylinder geometry. To completely construct A, the parameters needed are R, Z, dz, η, and Δt. (For compatibility with TAY, the actual equation being solved is equation (C.5), which has Δt on the left-hand side of the equation, and hence Δt is incorporated into A.) MATLAB has a built-in function, `pinv`, for computing the pseudo-inverse.

Once A^+ was computed by MATLAB, it was imported into the NPTAY and TAY programs via a text file. Tests were made to ensure that there was no roundoff error as a result of reading the entries A^+ from a text file rather than keeping A^+ in memory the entire time. The LAPACK routing DGELSS, which solves the least squares problem via the SVD as we do, was used on various Taylor-Couette geometries to verify this. The solution vectors returned by our method and by DGELSS agreed within 10^{-11}.

Using the iterative Gauss-Seidel method to solve (C.5) is a process which can take several hundred thousand iterations. Instead, we now solve the Poisson equation by a simple matrix-vector multiplication, a much faster process. The pressure p^{n+1} is found simply by computing $A^+(-\nabla \cdot \mathbf{v}^*)$, where, in a flagrant abuse of notation, the term $\nabla \cdot \mathbf{v}^*$ is meant to be interpreted as the vector obtained by approximating the divergence of $\mathbf{v}^*$ using second-order finite differences, evaluated at every gridpoint. As we have mentioned before, implementing this solver routine vastly sped up the NPTAY program, at the expense of having to store the entire matrix A^+ in memory.

Bibliography

[1] R. Adair and J. W. Thomas. Simulations of counter-rotating Taylor-Couette flow in short cylinders. In preparation, 2004.

[2] R. H. Adair. *Simulations of Taylor-Couette Flow.* PhD thesis, Colorado State University, 1997.

[3] G. Ahlers and D. S. Cannell. Wave-number selection in rotating Couette-Taylor flow. In E. Frehland, editor, *Synergetics—From Microscopic to Macroscopic Order*, pages 27–32, Berlin, 1994. Springer-Verlag.

[4] T. Alziary de Roquefort and G. Grillaud. Computation of Taylor vortex flow by a transient implicit method. *Comp. Fluids*, 6:259–269, 1978.

[5] C. D. Andereck, R. Dickman, and H. L. Swinney. New flows in a circular Couette system with co-rotating cylinders. *Phys. Fluids*, 26(6):1395–1401, June 1983.

[6] C. D. Andereck, S. S. Liu, and H. L. Swinney. Flow regimes in a circular Couette system with independently rotating cylinders. *J. Fluid Mech.*, 164:155–183, 1986.

[7] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK *Users' Guide.* SIAM, Philadelphia, third edition, 1999. LAPACK software available at `http://www.netlib.org/lapack`.

[8] D. K. Anson, T. Mullin, and K. A. Cliffe. A numerical and experimental investigation of a new solution in the Taylor vortex problem. *J. Fluid Mech.*, 207:475–487, 1989.

[9] A. Back, J. Guckenheimer, M. Myers, F. Wicklin, and P. Worfolk. DsTool: Computer assisted exploration of dynamical systems. *Notices Amer. Math. Soc.*, 39(4):303–309, April 1992. Software available at `http://www.cam.cornell.edu/~gucken/dstool`.

[10] T. B. Benjamin. Bifurcation phenomena in steady flows of a viscous fluid I: Theory. *Proc. R. Soc. Lond. A*, 359:1–26, 1978.

[11] T. B. Benjamin. Bifurcation phenomena in steady flows of a viscous fluid II: Experiments. *Proc. R. Soc. Lond. A*, 359:27–43, 1978.

[12] T. B. Benjamin and T. Mullin. Anomalous modes in the Taylor experiment. *Proc. R. Soc. Lond. A*, 377:221–249, 1981.

[13] T. B. Benjamin and T. Mullin. Notes on the multiplicity of flows in the Taylor experiment. *J. Fluid Mech.*, 121:219–230, 1982.

[14] A. Bergeon, D. Henry, H. BenHadid, and L. S. Tuckerman. Marangoni convection in binary mixtures with Soret effect. *J. Fluid Mech.*, 375:143–177, 1998.

[15] W.-J. Beyn, A. Champneys, E. Doedel, W. Govaerts, Y. A. Kuznetsov, and B. Sandstede. Numerical continuation, and computation of normal forms. In B. Fiedler, editor, *Handbook of Dynamical Systems*, volume 2, pages 149–219. Elsevier, 2002.

[16] J. H. Bolstad and H. B. Keller. Computation of anomalous modes in the Taylor experiment. *J. Comput. Phys.*, 69:230–251, 1987.

[17] J. Bošek and V. Janovský. A note on the Recursive Projection Method. *Z. Angew. Math. Mech.*, 77:S437–S440, 1997.

[18] J. E. Burkhalter and E. L. Koschmieder. Steady supercritical Taylor vortex flow. *J. Fluid Mech.*, 58(3):547–560, 1973.

[19] J. E. Burkhalter and E. L. Koschmieder. Steady supercritical Taylor vortices after sudden starts. *Phys. Fluids*, 17(11):1929–1935, November 1974.

[20] K. Burrage and J. Erhel. On the performance of various adaptive preconditioned GMRES strategies. *Numer. Linear Algebra Appl.*, 5:101–121, 1998.

[21] K. Burrage, J. Erhel, B. Pohl, and A. Williams. A deflation technique for linear systems of equations. *SIAM J. Sci. Comput.*, 19(4):1245–1260, 1998.

[22] A. R. Champneys, Y. A. Kuznetsov, and B. Sandstede. HOMCONT: An AUTO86 driver for homoclinic bifurcation analysis. Version 2.0. Technical Report AM-R9516, Centrum voor Wiskunde en Informatica, Amsterdam, 1995.

[23] S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability.* Oxford University Press, London, 1961.

[24] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22(104):745–762, 1968.

[25] P. Chossat and G. Iooss. *The Couette-Taylor Problem*, volume 102 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1994.

[26] K. A. Cliffe. Numerical calculations of two-cell and single-cell Taylor flows. *J. Fluid Mech.*, 135:219–233, 1983.

[27] K. A. Cliffe. Numerical calculations of the primary-flow exchange process in the Taylor problem. *J. Fluid Mech.*, 197:57–79, 1988.

[28] K. A. Cliffe. ENTWIFE (Release 6.3) Reference Manual: ENTWIFE, INITIAL DATA AND SOLVER DATA COMMANDS, 1996. See `http://www.entwife.com`.

[29] K. A. Cliffe, J. J. Kobine, and T. Mullin. The role of anomalous modes in Taylor-Couette flow. *Proc. R. Soc. Lond. A*, 439:341–357, 1992.

[30] K. A. Cliffe and T. Mullin. New results on two-cell Taylor flows. In C. Taylor, M. D. Olson, P. M. Gresho, and W. G. Habashi, editors, *Numerical Methods in Laminar and Turbulent Flow, Part 1, 2*, pages 152–163, Swansea, 1985. Pineridge Press.

[31] K. A. Cliffe and T. Mullin. A numerical and experimental study of anomalous modes in the Taylor experiment. *J. Fluid Mech.*, 153:243–258, 1985.

[32] K. A. Cliffe and A. Spence. The calculation of high order singularities in the finite Taylor problem. In T. Küpper, H. D. Mittelmann, and H. Weber, editors, *Numerical Methods for Bifurcation Problems*, volume 70 of *International Series of Numerical Mathematics*, pages 129–144. Birkhäuser, Basel-Boston-Berlin, 1984.

[33] J. A. Cole. Taylor-vortex instability and annulus-length effect. *J. Fluid Mech.*, 75:1–15, 1976.

[34] D. Coles. Transition in circular Couette flow. *J. Fluid Mech.*, 21(3):385–425, 1965.

[35] M. Couette. Études sur le frottement des liquides. *Ann. Chim. Phys.*, 6(21):433–510, 1890.

[36] K. T. Coughlin and P. S. Marcus. Modulated waves in Taylor-Couette flow. Part 2. Numerical simulation. *J. Fluid Mech.*, 234:19–46, 1992.

[37] B. D. Davidson. Large-scale continuation and numerical bifurcation for partial differential equations. *SIAM J. Numer. Anal.*, 34(5):2008–2027, 1997.

[38] A. Dhooge, W. Govaerts, and Y. A. Kuznetsov. MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs. *ACM Trans. Math. Software*, 29(2):141–164, 2003.

[39] R. C. Di Prima and H. L. Swinney. Instabilities and transition in flow between concentric rotating cylinders. In H. L. Swinney and J. P. Gollub, editors, *Hydrodynamic Instabilities and the Transition to Turbulence*. Springer-Verlag, 1981.

[40] N. Dinar and H. B. Keller. Computations of Taylor vortex flows using multigrid continuation methods. In C. C. Chao, S. A. Orszag, and W. Shyy, editors, *Recent Advances in Computational Fluid Dynamics*, volume 43 of *Lecture Notes in Engineering*, pages 191–262. Springer-Verlag, Berlin, 1989.

[41] E. Doedel, H. B. Keller, and J. P. Kernevez. Numerical analysis and control of bifurcation problems: (I) bifurcations in finite dimensions. *Int. J. Bifurcation Chaos*, 1(3):493–520, 1991.

[42] E. Doedel, H. B. Keller, and J. P. Kernevez. Numerical analysis and control of bifurcation problems: (II) bifurcations in infinite dimensions. *Int. J. Bifurcation Chaos*, 1(4):745–772, 1991.

[43] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Y. A. Kuznetsov, B. Sandstede, and X. Wang. AUTO*97: Continuation and Bifurcation Software for Ordinary Differential Equations (with* HOMCONT*)*. Concordia University, Montreal, Canada, March 1998. Software available at `http://indy.cs.concordia.ca/auto`.

[44] E. J. Doedel, W. Govaerts, and Y. A. Kuznetsov. Computation of periodic solution bifurcations in ODEs using bordered systems. *SIAM J. Numer. Anal.*, 41(2):401–435, 2003.

[45] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Software*, 14(1):1–17, 1988.

[46] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of FORTRAN basic linear algebra subprograms: Model implementation and test programs. *ACM Trans. Math. Software*, 14(1):18–32, 1988.

[47] R. J. Donnelly. Taylor-Couette flow: the early days. *Physics Today*, pages 32–39, November 1991.

[48] R. J. Donnelly. Evolution of instrumentation for Taylor-Couette flow. In C. D. Andereck and F. Hayot, editors, *Ordered and Turbulent Patterns in Taylor-Couette Flow*, volume 297 of *NATO Advanced Science Institutes Series B: Physics*, pages 1–27. Plenum Press, New York, 1992.

[49] W. S. Edwards, L. S. Tuckerman, R. A. Friesner, and D. C. Sorensen. Krylov methods for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 110:82–102, 1994.

[50] K. Engelborghs, K. Lust, and D. Roose. Direct computation of period doubling bifurcation points of large-scale systems of ODEs using a Newton-Picard method. *IMA J. Numer. Anal.*, 19:525–547, 1999.

[51] J. Erhel, K. Burrage, and B. Pohl. Restarted GMRES preconditioned by deflation. *J. Comput. Appl. Math.*, 69:303–318, 1996.

[52] B. Ermentrout. *Simulating, Analyzing and Animating Dynamical Systems: A Guide to* XPPAUT *for Researchers and Students*. SIAM, Philadelphia, 2002. XPPAUT available at `http://www.math.pitt.edu/~bard/xpp/xpp.html`.

[53] P. R. Fenstermacher, H. L. Swinney, and J. P. Gollub. Dynamical instabilities and the transition to chaotic Taylor vortex flow. *J. Fluid Mech.*, 94(1):103–128, 1979.

[54] C. W. Gear, I. G. Kevrekidis, and C. Theodoropoulos. "Coarse" integration/bifurcation analysis via microscopic simulators: micro-Galerkin methods. *Comp. Chem. Engrg.*, 26:941–963, 2002.

[55] G. H. Golub and C. F. van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, London, third edition, 1996.

[56] M. Gorman and H. L. Swinney. Spatial and temporal characteristics of modulated waves in the circular Couette system. *J. Fluid Mech.*, 117:123–142, 1982.

[57] W. Govaerts, J. Guckenheimer, and A. Khibnik. Defining functions for multiple Hopf bifurcations. *SIAM J. Numer. Anal.*, 34(3):1269–1288, 1997.

[58] W. J. F. Govaerts. *Numerical Methods for Bifurcations of Dynamical Equilibria.* SIAM, Philadelphia, 2000.

[59] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: A Practical Introduction.* SIAM, Philadelphia, 1998.

[60] Griewank and Reddien. The calculation of Hopf points by a direct method. *IMA J. Numer. Anal.*, 3:295–303, 1983.

[61] J. Hale and H. Koçak. *Dynamics and Bifurcations*, volume 3 of *Texts in Applied Mathematics.* Springer-Verlag, New York-Heidelberg-Berlin, 1991.

[62] C. Hoffmann and M. Lücke. Spiral vortices and Taylor vortices in the annulus between counter-rotating cylinders. In C. Egbers and G. Pfister, editors, *Physics of Rotating Fluids*, volume 549 of *Lecture Notes in Physics*, pages 55–66. Springer-Verlag, New York, 2000.

[63] G. Iooss and D. D. Joseph. *Elementary Stability and Bifurcation Theory.* Springer-Verlag, New York-Heidelberg-Berlin, 1980.

[64] H. Ito and A. Kumamoto. Locating fold bifurcation points using subspace shooting. *IEICE Trans. Fund.*, E81-A(9):1791–1797, 1998.

[65] H. Jarausch. Analyzing stationary and periodic solutions of systems of parabolic partial differential equations by using singular subspaces as reduced basis. *Math. Comput. Modelling*, 20(10–11):69–87, 1994.

[66] H. Jarausch and W. Mackens. Numerical treatment of bifurcation branches by adaptive condensation. In T. Küpper, H. D. Mittelmann, and H. Weber, editors, *Numerical Methods for Bifurcation Problems*, volume 70 of *International Series of Numerical Mathematics*, pages 296–309. Birkhäuser, Basel-Boston-Berlin, 1984.

[67] H. Jarausch and W. Mackens. Computing bifurcation diagrams for large nonlinear variational problems. In P. Deuflhard and B. Engquist, editors, *Large Scale Scientific Computing*, volume 7 of *Progress in Scientific Computing*. Birkhäuser, 1987.

[68] H. Jarausch and W. Mackens. Solving large nonlinear systems of equations by an adaptive condensation process. *Numer. Math.*, 50(6):633–653, 1987.

[69] A. Jepson and A. Spence. Folds in solutions of two parameter systems and their calculation. Part I. *SIAM J. Numer. Anal.*, 22(2):347–368, 1985.

[70] A. D. Jepson and A. Spence. Singular points and their computation. In T. Küpper, H. D. Mittelmann, and H. Weber, editors, *Numerical Methods for Bifurcation Problems*, volume 70 of *International Series of Numerical Mathematics*, pages 195–209. Birkhäuser, Basel-Boston-Berlin, 1984.

[71] A. D. Jepson and A. Spence. The numerical solution of nonlinear equations having several parameters I: Scalar equations. *SIAM J. Numer. Anal.*, 22(4):736–759, 1985.

[72] A. D. Jepson, A. Spence, and K. A. Cliffe. The numerical solution of nonlinear equations having several parameters. Part III: Equations with Z_2-symmetry. *SIAM J. Numer. Anal.*, 28(3):809–832, 1991.

[73] H. B. Keller. Numerical solution of bifurcation and nonlinear eigenvalue problems. In P. H. Rabinowitz, editor, *Applications of Bifurcation Theory*. Academic Press, New York, 1997.

[74] H. B. Keller. RPM—a remedy for instability. In S. Tavener and D. Estep, editors, *Collected Lectures on the Preservation of Stability Under Discretization*, volume 109 of *Proceedings in Applied Mathematics*, pages 185–196, Philadelphia, 2002. SIAM.

[75] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, and C. Theodoropoulos. Equation-free, coarse-grained multiscale computation: Enabling microscopic simulators to perform system-level analysis. *Comm. Math. Sci.*, 1(4):715–762, 2003.

[76] A. I. Khibnik, Y. A. Kuznetsov, V. V. Levitin, and E. V. Nikolaev. Continuation techniques and interactive software for bifurcation analysis of ODEs and iterated maps. *Phys. D*, 62:360–371, 1993.

[77] E. D. Koronaki, A. G. Boudouvis, and I. G. Kevrekidis. Enabling stability analysis of tubular reactor models using PDE/PDAE integrators. *Comp. Chem. Engng.*, 27:951–964, 2003.

[78] M. A. Krasnosel'skii, G. M. Vainikko, P. P. Zabreiko, Y. B. Rutitskii, and V. Y. Stetsenko. *Approximate Solution of Operator Equations*. Wolters-Noordhoff Publishing, Groningen, 1972.

[79] J. Krishnan, K. Engelborghs, M. Bär, K. Lust, D. Roose, and I. G. Kevrekidis. A computer-assisted study of pulse dynamics in anisotropic media. *Phys. D*, 154:85–110, 2001.

[80] Y. A. Kuznetsov and V. V. Levitin. CONTENT*: A multiplatform environment for analyzing dynamical systems.* Dynamical Systems Laboratory, CWI, Amsterdam, 1995–1997. Available at `http://ftp.cwi.nl/CONTENT`.

[81] R. B. Lehoucq. Implicitly restarted Arnoldi methods and subspace iteration. *SIAM J. Matrix Anal.*, 23(2):551–562, 2001.

[82] R. B. Lehoucq, D. C. Sorensen, and C. Yang. ARPACK *User's Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, Philadelphia, 1998. ARPACK software available at `http://www.caam.rice.edu/software/ARPACK`.

[83] A. Lorenzen and T. Mullin. Anomalous modes and finite-length effects in Taylor-Couette flow. *Phys. Rev. A*, 31(5):3463–3465, May 1985.

[84] P. Love. *Bifurcations in Kolmogorov and Taylor-Vortex Flow.* PhD thesis, California Institute of Technology, 1998.

[85] K. Lust. *Numerical Bifurcation Analysis of Periodic Solutions of Partial Differential Equations.* PhD thesis, Katholieke Universiteit Leuven, 1997.

[86] K. Lust. PDECONT 1.5: A timestepper-based continuation code for large-scale systems, 2003. Available at `http://www.cs.kuleuven.ac.be/~kurt/r_PDEcont.html`.

[87] K. Lust and D. Roose. Newton-Picard methods with subspace iteration for computing periodic solutions of partial differential equations. *Z. Angew. Math. Mech.*, 76:605–606, 1996. Supplement 2.

[88] K. Lust and D. Roose. Computation and bifurcation analysis of periodic solutions of large-scale systems. In E. Doedel and L. S. Tuckerman, editors, *Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems*, volume 119 of *IMA Volumes in Mathematics and its Applications.* Springer-Verlag, 2000.

[89] K. Lust, D. Roose, A. Spence, and A. R. Champneys. An adaptive Newton-Picard algorithm with subspace iteration for computing periodic solutions. *SIAM J. Sci. Comput.*, 19(4):1188–1209, 1998.

[90] T. Luzyanina, K. Engelborghs, K. Lust, and D. Roose. Computation, continuation and bifurcation analysis of periodic solutions of delay differential equations. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 7(11):2547–2560, 1997.

[91] A. G. Makeev, D. Maroudas, and I. G. Kevrekidis. "Coarse" stability and bifurcation analysis using stochastic simulators: Kinetic Monte Carlo examples. *J. Chem. Phys.*, 116(23):10083–10091, 2002.

[92] C. K. Mamun and L. S. Tuckerman. Asymmetry and Hopf bifurcation in spherical Couette flow. *Phys. Fluids*, 7(1):80–91, January 1995.

[93] P. S. Marcus. Simulation of Taylor-Couette flow. Part 1. Numerical methods and comparison with experiment. *J. Fluid Mech.*, 146:45–64, 1984.

[94] P. S. Marcus. Simulation of Taylor-Couette flow. Part 2. Numerical results for wavy-vortex flow with one travelling wave. *J. Fluid Mech.*, 146:65–113, 1984.

[95] K. A. Meyer. Time-dependent numerical study of Taylor vortex flow. *Phys. Fluids*, 10(9):1874–1879, 1967.

[96] K. A. Meyer. Three-dimensional study of flow between concentric rotating cylinders. *Phys. Fluids*, Supplement II:165–170, 1969.

[97] R. Meyer-Spasche. *Pattern Formation in Viscous Flows: The Taylor-Couette Problem and Rayleigh-Bénard Convection*, volume 128 of *International Series of Numerical Mathematics*. Birkhäuser, Basel-Boston-Berlin, 1999.

[98] R. Meyer-Spasche and H. B. Keller. Computations of the axisymmetric flow between rotating cylinders. *J. Comput. Phys.*, 35:100–109, 1980.

[99] R. Meyer-Spasche and H. B. Keller. Some bifurcation diagrams for Taylor vortex flows. *Phys. Fluids*, 28(5):1248–1252, May 1985.

[100] R. Meyer-Spasche and M. Wagner. Steady axisymmetric Taylor vortex flows with free stagnation points of the poloidal flow. In T. Küpper, R. Seydel, and H. Troger, editors, *Bifurcation: Analysis, Algorithms, Applications*, volume 79 of *International Series of Numerical Mathematics*, pages 213–221, Basel-Boston-Berlin, 1987. Birkhäuser.

[101] J. Möller, O. Runborg, P. G. Kevrekidis, K. Lust, and I. G. Kevrekidis. Effective equations for discrete systems: A time stepper based approach. *Submitted to Nonlinearity*, 2003. Also available as physics/0307153 at arXiv.org e-Print archive.

[102] G. Moore and A. Spence. The calculation of turning points of nonlinear equations. *SIAM J. Numer. Anal.*, 17(4):567–576, 1980.

[103] R. D. Moser, P. Moin, and A. Leonard. A spectral numerical method for the Navier-Stokes equations with applications to Taylor-Couette flow. *J. Comput. Phys.*, 52:524–544, 1983.

[104] T. Mullin and T. B. Benjamin. Transition to oscillatory motion in the Taylor experiment. *Nature*, 288:567–569, December 1980.

[105] T. Mullin, Y. Toya, and S. J. Tavener. Symmetry breaking and multiplicity of states in small aspect ratio Taylor-Couette flow. *Phys. Fluids*, 14(8):2778–2787, 2002.

[106] A. Nayfeh and B. Balachandran. *Applied Nonlinear Dynamics: Analytical, Computational, and Experimental Methods.* John Wiley & Sons, Inc., New York, 1995.

[107] G. P. Neitzel. Numerical computation of time-dependent Taylor-vortex flows in finite-length geometries. *J. Fluid Mech.*, 141:51–66, 1984.

[108] G. Pfister, H. Schmidt, K. A. Cliffe, and T. Mullin. Bifurcation phenomena in Taylor-Couette flow in a very short annulus. *J. Fluid Mech.*, 191:1–18, 1988.

[109] M. Poliashenko and C. K. Aidun. A direct method for computation of simple bifurcations. *J. Comput. Phys.*, 121:246–260, 1995.

[110] D. Roose and V. Hlavaček. A direct method for the computation of Hopf bifurcation points. *SIAM J. Appl. Math.*, 45(6):879–894, 1985.

[111] D. Roose, K. Lust, A. Champneys, and A. Spence. A Newton-Picard shooting method for computing periodic solutions of large-scale dynamical systems. *Chaos, Solitons & Fractals*, 5(10):1913–1925, 1995.

[112] O. Runborg, C. Theodoropoulos, and I. G. Kevrekidis. Effective bifurcation analysis: a time-stepper-based approach. *Nonlinearity*, 15:491–511, 2002.

[113] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, Manchester, 1992.

[114] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. D. Wilkes, E. A. Burroughs, R. B. Lehoucq, and L. A. Romero. LOCA *1.1 Library of Continuation Algorithms: Theory and Implementation Manual.* Sandia National Laboratories, Albuquerque, NM, October 2002. Available at `http://www.cs.sandia.gov/loca`.

[115] R. M. Samelson and C. L. Wolfe. A nonlinear baroclinic wave-mean oscillation with multiple normal mode instabilities. *J. Atmos. Sci.*, 60:1186–1199, 2003.

[116] D. G. Schaeffer. Qualitative analysis of a model for boundary effects in the Taylor problem. *Math. Proc. Camb. Phil. Soc.*, 87:307–337, 1980.

[117] W. Schröder and H. B. Keller. Wavy Taylor-vortex flows via multigrid continuation methods. *J. Comput. Phys.*, 91:197–227, 1990.

[118] A. Schulz and G. Pfister. Bifurcation and structure of flow between counter-rotating cylinders. In C. Egbers and G. Pfister, editors, *Physics of Rotating Fluids*, pages 37–54. Springer-Verlag, New York, 2000.

[119] A. Schulz, G. Pfister, and S. J. Tavener. The effect of outer cylinder rotation on Taylor-Couette flow at small aspect ratio. *Phys. Fluids*, 15(2):417–425, 2003.

[120] R. Seydel. *From Equilibrium to Chaos: Practical Bifurcation and Stability Analysis.* Elsevier, New York-Amsterdam-London, 1988.

[121] G. M. Shroff and H. B. Keller. Stabilization of unstable procedures: the Recursive Projection Method. *SIAM J. Numer. Anal.*, 30(4):1099–1120, 1993.

[122] H. A. Snyder. Stability of rotating Couette flow. I. Asymmetric waveforms. *Phys. Fluids*, 11(4):728–734, April 1968.

[123] H. A. Snyder. Stability of rotating Couette flow. II. Comparison with numerical results. *Phys. Fluids*, 11(8):1599–1605, August 1968.

[124] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal.*, 13(1):357–385, 1992.

[125] H. Specht, M. Wagner, and R. Meyer-Spasche. Interactions of secondary branches of Taylor vortex solutions. *Z. Angew. Math. Mech.*, 69(10):339–352, 1989.

[126] A. Spence and A. D. Jepson. The numerical calculation of cusps, bifurcation points and isola formation points in two parameter problems. In T. Küpper, H. D. Mittelmann, and H. Weber, editors, *Numerical Methods for Bifurcation Problems*, volume 70 of *International Series of Numerical Mathematics*, pages 502–514. Birkhäuser, Basel-Boston-Berlin, 1984.

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

[128] C. L. Streett and M. Y. Hussaini. A numerical simulation of the appearance of chaos in finite-length Taylor-Couette flow. *Appl. Numer. Math.*, 7:41–71, 1991.

[129] S. H. Strogatz. *Nonlinear Dynamics and Chaos*. Perseus Publishing, Cambridge, MA, 1994.

[130] S. J. Tavener. Personal communication, October 2001.

[131] S. J. Tavener, T. Mullin, and K. A. Cliffe. Novel bifurcation phenomena in a rotating annulus. *J. Fluid Mech.*, 229:483–497, 1991.

[132] G. I. Taylor. Stability of a viscous liquid contained between two rotating cylinders. *Phil. Trans. R. Soc. London, Ser. A*, 223:289–343, 1923.

[133] R. Témam. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionaires. *Arch. Rational Mech. Anal.*, 32:135–153, 1969.

[134] C. Theodoropoulos, Y.-H. Qian, and I. G. Kevrekidis. "Coarse" stability and bifurcation analysis using time-steppers: A reaction-diffusion example. *Proc. Natl. Acad. Sci. USA*, 97(18):9840–9843, 2000.

[135] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*, volume 22 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 1995.

[136] J. W. Thomas. *Numerical Partial Differential Equations: Conservation Laws and Elliptic Equations*, volume 33 of *Texts in Applied Mathematics*. Springer-Verlag, New York, 1999.

[137] G. Tiesinga, F. W. Wubs, and A. E. P. Veldman. Bifurcation analysis of incompressible flow in a driven cavity by the Newton-Picard method. *J. Comput. Appl. Math.*, 140:751–772, 2002.

[138] L. N. Trefethen and D. Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[139] L. S. Tuckerman and D. Barkley. Bifurcation analysis for timesteppers. In E. Doedel and L. S. Tuckerman, editors, *Numerical Methods for Bifurcation Problems and Large-Scale Dynamical Systems*, volume 119 of *IMA Volumes in Mathematics and its Applications*. Springer-Verlag, 2000.

[140] M. Valorani and D. A. Goussis. Explicit time-scale splitting algorithm for stiff problems: Auto-ignition of gaseous mixtures behind a steady shock. *J. Comput. Phys.*, 169:44–79, 2001.

[141] T. L. van Noorden, S. M. V. Lunel, and A. Bliek. The efficient computation of periodic states of cyclically operated chemical processes. *IMA J. Appl. Math.*, 68:149–166, 2003.

[142] C. Vuik, A. Segal, and J. A. Meijerink. An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. *J. Comput. Phys.*, 152:385–403, 1999.

[143] B. Werner. Computation of Hopf bifurcation with bordered matrices. *SIAM J. Numer. Anal.*, 33(2):435–455, 1996.

[144] B. Werner and V. Janovsky. Computation of Hopf branches bifurcating from Takens-Bogdanov points for problems with symmetries. In R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, editors, *Bifurcation and Chaos: Analysis, Algorithms, Applications*, volume 97 of *International Series of Numerical Mathematics*, pages 377–388. Birkhäuser, Basel-Boston-Berlin, 1991.

[145] B. Werner and A. Spence. The computation of symmetry-breaking bifurcation points. *SIAM J. Numer. Anal.*, 21(2):388–399, 1984.

[146] K. H. Winters, K. A. Cliffe, and C. P. Jackson. The prediction of instabilities using bifurcation theory. In R. W. Lewis, E. Hinton, P. Bettress, and B. A. Schrefler, editors, *Numerical Methods for Transient and Coupled Problems*, Wiley Series in Numerical Methods in Engineering, chapter 9, pages 179–198. John Wiley & Sons, 1987.