JOURNAL OF COMPUTATIONAL PHYSICS 94, 382–402 (1991)

Numerical Solution of Eigenvalue Problems for Linear Boundary Value ODEs*

S. BRAMLEY

Department of Mathematics,
University of Strathclyde, Glasgow G1 1XH, United Kingdom

L. DIECI†

School of Mathematics,
Georgia Institute of Technology, Atlanta, Georgia 30332

AND

R. D. RUSSELL‡

Department of Mathematics & Statistics,
Simon Fraser University, Burnaby, British Colombia, Canada V5A 1S6

Received June 12, 1989; revised January 19, 1990

Interrelationships between several popular approaches for solving eigenvalue problems for linear boundary value ODEs are given. For linear eigenvalue problems, the popular methods can be interpreted in a common framework. This leads us to propose and justify alternative strategies. The choice of numerical methods used here is motivated by the desire to solve eigenvalue problems for stiff ODEs. In particular, we consider a one-step global method (spline collocation) and two initial value methods (Riccati and continuous orthonormalization) to solve the Orr–Sommerfeld equation. A comparison of results for these methods, using various implementation strategies, is given. © 1991 Academic Press, Inc.

I. INTRODUCTION

Many methods for computing eigenvalues for boundary value problems for ordinary differential equations (BVODEs) are presented in the literature, but frequently little discussion is given of their relative merits. There are of course

* This work was supported in part under NSERC (Canada) Grant A8781.
† Supported in part under NSF Grant DMS8802762.
‡ Supported in part under an SERC Fellowship at Imperial College, London.

0021-9991/91 $3.00

several reasons for this. Often the methods are tailor-made for a specific class of problems (e.g., Sturm–Liouville problems [7]). Also, it can be notoriously difficult to make definitive statements about the relative merits of one method versus another in a general setting. Nevertheless, we feel that it is possible, and important, to compare some of the more promising methods at this time.

Here, we limit the setting by concentrating on the problem of computing to high accuracy a small number of eigenvalues of the Orr–Sommerfeld equation for large Reynolds number. The fundamental importance of this problem is well known and more than adequately described elsewhere, e.g., see [17]. For Orr–Sommerfeld-type problems, eigenfunctions with boundary layers, occurring because of fast fundamental solution components, must be resolved. The most successful general methods to date for solving this type of problem in the standard BVODE context are the so-called *global methods* collocation and finite differences [4], although initial value (IV) methods suited for handling the fast components also deserve close consideration. In particular, we feel that a Riccati-type method [13] and continuous orthonormalization [1, 8, 12, 23, 21] are worth further attention. In this paper, we report on a spline collocation method, as the representative of (and competitive with the other) global methods, and these two IV methods.

Even *given* that these are the methods one wishes to consider, the difficulty of any comparison would be compounded by the fact that there are several general strategies for computing eigenvalues and various ways to implement them. While this makes detailed comparison of questionable value, we believe that our results are a faithful indication of the potential and feasibility of the methods and of the different strategies.

We use some fairly highly developed software for spline collocation [6], and in this respect some of our presentation is similar in spirit to that in [3], where the capability of collocation software for solving standard BVODEs was reported. Comparable software is not as yet available for the initial value approaches, so we use our own unsophisticated implementations of the Riccati and continuous orthonormalization methods.

An outline of the paper is as follows: In Section 2, the main approaches for computing eigenvalues are delineated and a common framework allowing interpretation of the interrelationship between them is given. The basic implementations of numerical methods are discussed in Section 3. Numerical results for the Orr–Sommerfeld equation are given in Section 4, followed by conclusions in Section 5.

II. The Eigenvalue Problem

We consider the following linear eigenvalue problem for a linear BVODE with separated boundary conditions (BCs):

$$\mathbf{y}' = (\lambda C(t) + D(t))\,\mathbf{y}, \qquad 0 < t < 1 \tag{1a}$$

$$B_1\mathbf{y}(0) = 0_p, \qquad B_2\mathbf{y}(1) = 0_q, \tag{1b}$$

where $\mathbf{y}(t) = (y_1(t), ..., y_n(t))^T \in \mathbf{C}^n$, 0_p and 0_q are zero p- and q-vectors, respectively, $B_1 \in \mathbf{C}^{p \times n}$, $B_2 \in \mathbf{C}^{q \times n}$ $(p+q=n)$, and $\lambda \in \mathbf{C}$. Extending the results for nonseparated BCs is generally straightforward [4]. Also, while we consider the linear eigenvalue problem in the "standard" form (1a), (1b), most of the techniques and conclusions carry over to the more general problem $\mathbf{y}' = A(t, \lambda)\,\mathbf{y}$ (which allows for a nonlinear dependence on λ).

For fixed λ, if $Y(t, \lambda)$ is a (nonsingular) fundamental solution matrix for (1a), then $\mathbf{y}(t, \lambda) = Y(t, \lambda)\,\boldsymbol{\alpha}$ solves (1a), (1b) iff $Q(\lambda)\,\boldsymbol{\alpha} = 0$, where

$$Q(\lambda) := \begin{bmatrix} B_1 \\ 0 \end{bmatrix} Y(0, \lambda) + \begin{bmatrix} 0 \\ B_2 \end{bmatrix} Y(1, \lambda) =: \bar{B}_1\, Y(0, \lambda) + \bar{B}_2\, Y(1, \lambda). \tag{2}$$

Thus, λ^* is an eigenvalue of (1a), (1b) iff $Q(\lambda^*)$ is singular.

This relationship is at the core of the first type of procedure to calculate λ^*:

Approach 1. An iterative process is used to determine a root λ^* of a function $f(\lambda)$ whose zeros coincide with points where $Q(\lambda)$ is singular.

For initial value techniques, this is frequently done by essentially finding $Y(t, \lambda)$ and using a root finder to solve $f(\lambda) := \det(Q(\lambda)) = 0$. It has been a standard approach for IV methods and has met with good success [11, 12, 23]. Also, eigenvalues of multiplicity greater than one can be treated with this approach. Using $f(\lambda) = \sigma_1((Q(\lambda))$ instead, where $\sigma_1((Q(\lambda))$ is the smallest singular value of $Q(\lambda)$ (computed say from the SVD), would offer improved numerical stability and a reliable estimate of multiplicity. However, $f(\lambda)$ would not undergo a sign change since $\sigma_1 \geqslant 0$, so the method could need modification (e.g., replace an interval type root finder by a suitable descent algorithm).

For global methods, the algorithms using Approach 1 consist of globally discretizing the ODE over $[0, 1]$, and then solving a matrix eigenvalue problem for the full discretization matrix, where one argues that the eigenvalues of the full system approach those for the original BVODE. Spectral methods, where approximate eigenfunctions are expressed in terms of a series of orthogonal polynomials or trigonometric functions [26] are in fact of this type. Several spline collocation and finite difference methods of this type have also been used successfully [9, 28]. Typically, with this approach one computes a large number of approximate eigenvalues, but high accuracy can be difficult to obtain efficiently. With some spectral methods, including the one we use here, spurious eigenvalues (corresponding to the discretized system but not to the original differential system) can arise, although modified versions which eliminate such spurious modes have recently been considered (e.g., see [16]). In our context, where a small number of eigenvalue approximations are sought, the next approach seems considerably more efficient [18] (although spectral methods can nevertheless be put to essential use by providing accurate initial approximations—see Section 4).

Approach 2. The eigenvalue problem (1a), (1b) is expanded to a standard BVODE for which an eigenvalue and eigenfunction correspond to an isolated solution, for which this enlarged problem is solved numerically.

This BVODE can be constructed in a variety of ways. One is to add the artificial ODE $\lambda' = 0$ and some type of normalization for the eigenfunction [18]. The simplest normalization occurs if one component of the eigenfunction is known to be nonzero at an endpoint. For example, with suitable permutation of variables if necessary, one could add the auxilliary conditions

$$\lambda' = 0, \qquad y_1(1) = 1 \tag{1c}$$

to (1a), (1b). Another normalization is to force $\int_0^1 \mathbf{y}^H \mathbf{y}\, dt = 1$ (where $\mathbf{y}^H$ is the conjugate transpose $\bar{\mathbf{y}}^T$) by using

$$\begin{aligned} \lambda' &= 0, & w(0) &= 0 \\ w' &= \mathbf{y}^H \mathbf{y}, & w(1) &= 1. \end{aligned} \tag{1d}$$

In general, it is not easy to find a suitable normalization or extra condition, though one can often be selected on physical grounds.

In any case, the enlarged ODE system is nonlinear even though the original eigenvalue problem is a linear one. Also, only implicit use is made of the fact that λ is constant. Nevertheless, the underlying motivation is to obtain a new problem for which one can use standard implementations of BVODE methods, including the robust software for global methods. An initial approximation λ_0 for the eigenvalue *and* $\mathbf{y}_0$ for the eigenfunction are now required. If only λ_0 is available, it can often be used to define a standard BVODE using (1a), (1b), except with some type of nonhomogeneous BC, e.g., with the normalizing condition of (1c) replacing one of the BCs in (1b), from which a corresponding $\mathbf{y}_0$ can be computed. All of the standard convergence theory now goes through when the eigenpair $(\lambda^*, \mathbf{y}^*)$ is an isolated solution to (1a), (1b), (1c), or (1a), (1b), (1d), so λ^* must be simple.

The next approach which we consider bears much in common with the previous one, except that the disadvantage of having an artificially enlarged system is overcome.

Approach 3. The original boundary value problem (1a), (1b) is discretized over $[0, 1]$, and the resulting eigensystem for $(\lambda^*, \mathbf{y}^*)$ is solved, explicitly treating λ as a parameter.

In other words, the discrete system is augmented by the single variable λ. This approach is used in [20] with a finite difference discretization method to handle BVPs with parameters. As we shall see, its solution still requires adding an extra condition, either implicitly or as an explicit normalization.

These three approaches to the eigenvalue problem have common features, as is seen by considering Approach 2 in some detail: For simplicity, consider the

BVODE (1a), (1b), (1c) and its solution by the process of quasilinearization. First, rewrite the problem as

$$\mathbf{z}' := \begin{bmatrix} \mathbf{y} \\ \lambda \end{bmatrix}' = \begin{bmatrix} (\lambda C(t) + D(t))\,\mathbf{y} \\ 0 \end{bmatrix} =: \mathbf{f}(t, \mathbf{z}), \qquad 0 < t < 1 \tag{3a}$$

$$B_1 \mathbf{y}(0) = 0_p, \qquad B_2 \mathbf{y}(1) = 0_q, \qquad y_1(1) = 1. \tag{3b}$$

Given an initial approximation $\begin{bmatrix} \mathbf{y}_0 \\ \lambda_0 \end{bmatrix}$, linearization gives the successive problems

$$\mathbf{y}' = (\lambda_i C(t) + D(t))\,\mathbf{y} + (\lambda - \lambda_i)\, C(t)\, \mathbf{y}_i(t) \tag{4a}$$

$$\lambda' = 0 \tag{4b}$$

$$B_1 \mathbf{y}(0) = 0_p, \qquad B_2 \mathbf{y}(1) = 0_q, \qquad y_1(1) = 1 \tag{4c}$$

to be solved for $\mathbf{y}_{i+1} = \mathbf{y}$, $\lambda_{i+1} = \lambda$, for $i = 0, 1, 2, \ldots$. This shows how the ODE for the approximation to $\mathbf{y}^*$ changes during the quasilinearization process. By writing (4a) as

$$\mathbf{y}' = (\lambda_i C(t) + D(t))\,\mathbf{y} + \mathbf{g}_i(t), \tag{5a}$$

where

$$\mathbf{g}_i(t) = (\lambda - \lambda_i)\, C\mathbf{y}_i(t), \tag{5b}$$

the ODE can be interpreted as one with an inhomogeneity involving λ_{i+1}, which approaches zero as $\lambda_i \to \lambda^*$.

It is instructive to see how the update λ_{i+1} is determined through the BCs (4c). For this, let

$$\mathbf{w} := \begin{bmatrix} \mathbf{y}_{i+1} \\ \lambda_{i+1} \end{bmatrix}, \qquad E := \begin{bmatrix} (\lambda_i C + D) & C\mathbf{y}_i \\ 0^{\mathrm{T}} & 0 \end{bmatrix}, \qquad \mathbf{q} := \begin{bmatrix} -\lambda_i C\mathbf{y}_i \\ 0 \end{bmatrix},$$

so (4a), (4b) becomes

$$\mathbf{w}' = E(t)\,\mathbf{w} + \mathbf{q}, \qquad 0 < t < 1. \tag{6}$$

Because of the structure of E, we can consider a fundamental solution for $\mathbf{w}' = E\mathbf{w}$ of the form

$$W(t) = \begin{bmatrix} Y(t) & \mathbf{w}_p(t) \\ 0_n^{\mathrm{T}} & 1 \end{bmatrix}, \tag{7}$$

where $Y(t) \in \mathbf{C}^{n \times n}$ is a fundamental solution for (5a) (or (1a) with $\lambda = \lambda_i$) and $\mathbf{w}_p(t)$ is a particular solution to (5a) with $\mathbf{g}_i = C\mathbf{y}_i$. Then, the solution to (6) is

$$\mathbf{w} = W(t)\,\gamma + \begin{bmatrix} \mathbf{v}(t) \\ 0 \end{bmatrix}, \tag{8}$$

where $\mathbf{v}(t)$ is a particular solution to (5a) with $\mathbf{g}_i = -\lambda_i C\mathbf{y}_i$. The BCs (4c) determine $\gamma = \begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix} \in \mathbf{C}^{n+1}$, where $\lambda_{i+1} = \gamma_2$. Specifically (with $\mathbf{e}_1 := (1, 0, ..., 0)^{\mathrm{T}}$),

$$\left\{\begin{bmatrix} B_1 & 0_p \\ 0 & 0_q \\ 0_n^{\mathrm{T}} & 0 \end{bmatrix}\begin{bmatrix} Y(0) & \mathbf{w}_p(0) \\ 0_n^{\mathrm{T}} & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0_p \\ B_2 & 0_q \\ \mathbf{e}_1^{\mathrm{T}} & 0 \end{bmatrix}\begin{bmatrix} Y(1) & \mathbf{w}_p(1) \\ 0_n^{\mathrm{T}} & 1 \end{bmatrix}\right\}\begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix}$$

$$= \begin{bmatrix} 0_n \\ 1 \end{bmatrix} - \begin{bmatrix} B_1\mathbf{v}(0) \\ 0_q \\ 0 \end{bmatrix} - \begin{bmatrix} 0_p \\ B_2\mathbf{v}(1) \\ \mathbf{e}_1^{\mathrm{T}}\mathbf{v}(1) \end{bmatrix}$$

or from (2)

$$Q(\lambda_i)\gamma_1 = \begin{bmatrix} B_1 Y(0) \\ B_2 Y(1) \end{bmatrix}\gamma_1 = -\begin{bmatrix} B_1\mathbf{v}(0) \\ B_2\mathbf{v}(1) \end{bmatrix} - \gamma_2\begin{bmatrix} B_1\mathbf{w}_p(0) \\ B_2\mathbf{w}_p(1) \end{bmatrix} \tag{9a}$$

$$\mathbf{e}_1^{\mathrm{T}} Y(1)\gamma_1 + w_1(1)\gamma_2 = 1 - v_1(1) \tag{9b}$$

(with $\mathbf{w}_p = (w_1, w_2, ..., w_n)^{\mathrm{T}}$, $\mathbf{v} = (v_1, v_2, ..., v_n)^{\mathrm{T}}$). To summarize, we have the following type of algorithm arising for Approach 2, solving (4a), (4b), (4c) with quasilinearization:

TYPE 2 ALGORITHM. Given an initial approximation $\begin{bmatrix}\mathbf{y}_0\\ \lambda_0\end{bmatrix}$. For $i = 0, 1, 2, ...,$

1. Compute (approximately) a fundamental solution $W(t)$ and particular solution $\begin{bmatrix}\mathbf{v}(t)\\ 0\end{bmatrix}$ for (6).

2. Solve (9a), (9b) for $\gamma = \begin{bmatrix}\gamma_1\\ \gamma_2\end{bmatrix}$ and let $\begin{bmatrix}\mathbf{y}_{i+1}\\ \lambda_{i+1}\end{bmatrix} = W(t)\gamma + \begin{bmatrix}\mathbf{v}(t)\\ 0\end{bmatrix}$.

As $\lambda_i \to \lambda^*$, $Q(\lambda_i)$ becomes singular, but (9b) provides the extra condition for solving a well-posed system if (1a), (1b), (1c) is, thereby giving $\begin{bmatrix}\gamma_1^*\\ \gamma_2^*\end{bmatrix}$ corresponding to an eigenfunction so long as $\operatorname{rank}(Q(\lambda^*)) = n - 1$. This again highlights the requirement that λ^* be a simple eigenvalue.

Thus, we see how quasilinearization for the enlarged system (1a), (1b), (1c) explicitly involves the matrix $Q(\lambda)$. It can be shown that the common numerical methods—one step finite difference and spline collocation methods, stabilized march and multiple shooting, the Riccati method, and continuous orthonormalization—can all be interpreted as schemes which determine approximations to appropriate fundamental solution matrices and particular solutions. For global methods this interpretation is done by assuming a sufficiently fine mesh, where (9a) is replaced by the full discretization matrix, while for initial value methods the interpretation is more direct since they are conceptually exact methods (i.e., their very formulation is done in terms of computing the exact solutions of ODEs). What distinguishes the various methods from each other in practice involves the details of the process such as how much explicit computation of $W(t)$ is done and how well scaled that particular fundamental solution matrix is [19, 4]. Regardless, Type 2

algorithms for the various methods are interpretable (at least in the limit) as approximations to the continuous process giving (8), (9).

It is generally possible to modify these algorithms such that at each iteration one dispenses with the ODE $\lambda' = 0$ altogether, but still uses the extra normalizing BC to determine $\begin{bmatrix} \mathbf{y}_{i+1} \\ \lambda_{i+1} \end{bmatrix}$. Then, the implementation can be classified as Approach 3. A natural way to do this is as follows: Note that $\bar{\mathbf{y}}_p(t) := \mathbf{v}(t) + \gamma_2 \mathbf{w}_p(t)$ is a particular solution to (4a). While $\bar{\mathbf{y}}_p(t)$ cannot be computed directly because λ_{i+1} is unknown, one can compute a fundamental solution $Y(t)$ and a particular solution $\mathbf{y}_p(t)$ for the ODE

$$\mathbf{y}' = (\lambda_i C(t) + D(t))\,\mathbf{y} + k_i C(t)\,\mathbf{y}_i(t), \qquad 0 < t < 1, \tag{10}$$

where the constant k_i is assumed for now to be some a priori estimate of $\lambda_{i+1} - \lambda_i$. There is no restriction in theory in assuming that $\mathbf{w}_p(t)$ and $\mathbf{v}(t)$ satisfy the same BCs, in which case $\mathbf{v}(t) = -\lambda_i \mathbf{w}_p(t)$ and $\bar{\mathbf{y}}_p(t) = (\lambda_{i+1} - \lambda_i)\,\mathbf{w}_p(t)$. So requiring that $\mathbf{y}_p(t)$ also satisfies the same BCs, $\bar{\mathbf{y}}_p(t) = ((\lambda_{i+1} - \lambda_i)/k_i)\,\mathbf{y}_p(t)$, and (9) can be expressed in terms of $\mathbf{y}_p(t)$. In particular, we have the following:

TYPE 3 Algorithm. Suppose that $(\lambda_0, \mathbf{y}_0)$ are given. For $i = 0, 1, 2, \ldots$,

1. Choose a constant k_i and compute a fundamental solution $Y(t)$ and a particular solution $\mathbf{y}_p(t)$ for (10).

2. Solve

$$Q(\lambda_i)\,\gamma_1 = -\frac{\lambda_{i+1} - \lambda_i}{k_i} \begin{bmatrix} B_1 \mathbf{y}_p(0) \\ B_2 \mathbf{y}_p(1) \end{bmatrix} \tag{11a}$$

$$\mathbf{e}_1^{\mathrm{T}} Y(1)\,\gamma_1 = 1 - \frac{\lambda_{i+1} - \lambda_i}{k_i}\,\mathbf{e}_1^{\mathrm{T}} \mathbf{y}_p(1) \tag{11b}$$

for $\begin{bmatrix} \gamma_1 \\ \lambda_{i+1} \end{bmatrix}$ and then set $\mathbf{y}_{i+1} = Y(t)\,\gamma_1 + ((\lambda_{i+1} - \lambda_i)/k_i)\,\mathbf{y}_p(t)$.

Since this is mathematically equivalent to an exact Type 2 algorithm, it is quadratically convergent if the problem is sufficiently smooth and $(\mathbf{y}_0, \lambda_0)$ is sufficiently close to a simple eigenpair $(\mathbf{y}^*, \lambda^*)$. Since the algorithm is independent of k_i, steps 1 and 2 of the algorithm simplify accordingly to:

1′. Compute $Y(t)$ and $\mathbf{y}_p(t)$ for (10) with $k_i = 1$.

2′. Solve

$$\begin{bmatrix} Q(\lambda_i) & \begin{bmatrix} B_1 \mathbf{y}_p(0) \\ B_2 \mathbf{y}_p(1) \end{bmatrix} \\ \mathbf{e}_1^{\mathrm{T}} Y(1) & \mathbf{e}_1^{\mathrm{T}} \mathbf{y}_p(1) \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \lambda_{i+1} \end{bmatrix} = \begin{bmatrix} \lambda_i \begin{bmatrix} B_1 \mathbf{y}_p(0) \\ B_2 \mathbf{y}_p(1) \end{bmatrix} \\ 1 + \lambda_i \mathbf{e}_1^{\mathrm{T}} \mathbf{y}_p(1) \end{bmatrix} \tag{11′}$$

and let $\mathbf{y}_{i+1} = Y(t)\,\gamma_1 + (\lambda_{i+1} - \lambda_i)\,\mathbf{y}_p(t)$.

These are used shortly in our developing a Type 3 Riccati method below.

It is natural to ask what advantage Approach 2 or 3 has over the traditional Approach 1 for an IV method such as multiple shooting. Since the majority of work for any approach involves computing the fundamental solution $Y(t)$ (and thus essentially the matrix $Q(\lambda_i)$), the extra computing cost for these approaches is not generally excessive. However, storage can be significantly greater than for Approach 1 if an eigenfunction approximation is not explicitly needed, since they still require keeping a global approximation for $\mathbf{y}_i(t)$ at each iteration (when integrating (4a) or (10)). On the other hand, computing a global approximation $\mathbf{y}_i(t)$ and then solving (4), these more complicated approaches typically have more robustness than does just using the simple shooting process to force $\det(Q(\lambda))=0$. (Verification that this can be the case is in Section 4.) Finally, note that Approaches 2 and 3 extend easily for the case where the original BVP is nonlinear.

III. Numerical Methods

If a numerical method is to be competitive for solving BVODEs having both fast increasing and fast decreasing fundamental solution components, it must be able to separate, or *decouple*, these two sets, and to invest its greatest labour in approximating them in regions where solution layers have the potential of occurring [4]. Riccati and continuous orthonormalization algorithms are designed to do this. Standard IV methods like simple shooting and stabilized march, which involve integrating the unmodified ODE (1a), generally do not, although implementations of them like the shooting codes in NAG and the code SUPORT [27] can of course be very efficient if fast components are not severe.

The Riccati method for solving general BVODEs which is implemented here is described in [13, 14], so we just consider how it adapts as a Type 3 algorithm to solve the enlarged (standard) BVODE (4). (For brevity we do not discuss the re-imbedding strategy because it turns out to be unnecessary for the numerical results in Section 4.) We use the block notation $\lambda_i C(t)+D(t)=:\left[\begin{smallmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{smallmatrix}\right]$, $C(t)\,\mathbf{y}_i=:\left[\begin{smallmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{smallmatrix}\right]$, $\mathbf{y}=\left[\begin{smallmatrix}\mathbf{z}\\ \mathbf{w}\end{smallmatrix}\right]$, where $A_{11}\in\mathbf{C}^{q\times q}$, $\mathbf{c}_1,\mathbf{z}\in\mathbf{C}^q$. If $\boldsymbol{\eta}_1=\left[\begin{smallmatrix}\mathbf{z}\\ \lambda\end{smallmatrix}\right]$, $\boldsymbol{\eta}_2=\mathbf{w}$ then (4) can be rewritten as

$$\begin{bmatrix}\boldsymbol{\eta}_1\\ \boldsymbol{\eta}_2\end{bmatrix}'=\begin{bmatrix}\mathbf{z}\\ \lambda\\ \mathbf{w}\end{bmatrix}'=\begin{bmatrix}A_{11} & \mathbf{c}_1 & A_{12}\\ 0_q^{\mathrm{T}} & 0 & 0_p^{\mathrm{T}}\\ A_{21} & \mathbf{c}_2 & A_{22}\end{bmatrix}\begin{bmatrix}\mathbf{z}\\ \lambda\\ \mathbf{w}\end{bmatrix}-\begin{bmatrix}\lambda_i\mathbf{c}_1\\ 0\\ \lambda_i\mathbf{c}_2\end{bmatrix} \tag{12a}$$

$$[B_{11}\quad 0_p\,|\,B_{12}]\begin{bmatrix}\boldsymbol{\eta}_1(0)\\ \boldsymbol{\eta}_2(0)\end{bmatrix}=0_p,\qquad \begin{bmatrix}B_{21} & 0_q & B_{22}\\ \mathbf{e}_1^{\mathrm{T}} & 0 & 0_2^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\eta}_1(1)\\ \boldsymbol{\eta}_2(1)\end{bmatrix}=\begin{bmatrix}0_q\\ 1\end{bmatrix}. \tag{12b}$$

Assume that the variables have been ordered so that $B_{12}\in\mathbf{C}^{p\times p}$ is invertible. (This may result in the normalizing condition $z_1(1)=1$ involving a different component after reordering, but it is straightforward to modify the algorithm to

handle such an eventuality.) A Riccati matrix $[R|\mathbf{r}] \in \mathbf{C}^{p\times(q+1)}$ and vector $\mathbf{x}$ such that

$$\boldsymbol{\eta}_2 = [R|\mathbf{r}]\,\boldsymbol{\eta}_1 + \mathbf{x} \tag{13}$$

are found by solving the IVPs [13, Eqs. (3.9)–(3.10)]:

$$[R|\mathbf{r}]' = [A_{21}|\mathbf{c}_2] + A_{22}[R|\mathbf{r}] - [R|\mathbf{r}]\begin{bmatrix} A_{11} & \mathbf{c}_1 \\ 0_q^{\mathrm{T}} & 0 \end{bmatrix} - [R|\mathbf{r}]\begin{bmatrix} A_{12} \\ 0_p^{\mathrm{T}} \end{bmatrix}[R|\mathbf{r}] \tag{14a}$$

$$[R(0)|\mathbf{r}(0)] = -B_{12}^{-1}[B_{11}|0_p], \tag{14b}$$

$$\mathbf{x}' = [A_{22} - RA_{12}]\,\mathbf{x} - \lambda_i(\mathbf{c}_2 - R\mathbf{c}_1) \tag{15a}$$

$$\mathbf{x}(0) = 0. \tag{15b}$$

Thus,

$$R' = A_{21} + A_{22}R - RA_{11} - RA_{12}R \tag{16a}$$

$$R(0) = -B_{12}^{-1}B_{11} \tag{16b}$$

$$\mathbf{r}' = [A_{22} - RA_{12}]\,\mathbf{r} + (\mathbf{c}_2 - R\mathbf{c}_1) \tag{17a}$$

$$\mathbf{r}(0) = 0, \tag{17b}$$

so $\mathbf{x}(t) = -\lambda_i\mathbf{r}(t)$ is directly available and integrating (15) unnecessary. From [13, Eq. (3.11)], after simplifying, we find that

$$\boldsymbol{\eta}_1' = \begin{bmatrix} \mathbf{z}' \\ \lambda' \end{bmatrix} = \begin{bmatrix} [A_{11} + A_{12}R]\,\mathbf{z} + (\lambda - \lambda_i)(A_{12}(t)\,\mathbf{r}(t) + \mathbf{c}_1(t)) \\ 0 \end{bmatrix}, \tag{18a}$$

$$[B_{21} + B_{22}R(1)]\,\mathbf{z}(1) + (\lambda - \lambda_i)\,B_{22}\mathbf{r}(1) = 0, \qquad z_1(1) = 1. \tag{18b}$$

Thus, the Riccati Type 2 algorithm involves solving (14), (17), and then (18) before the updated solution is formed via (13), but there is the following obvious modification:

RICCATI TYPE 3 ALGORITHM. Given an initial approximation $(\lambda_0, \mathbf{y}_0)$. For $i = 0, 1, 2, \ldots,$

1. Solve the IVPs (16), (17) for $R(t)$ and $\mathbf{r}(t)$ on $[0, 1]$.
2. Solve (18b) for $\begin{bmatrix} \mathbf{z}(1) \\ \lambda_{i+1} \end{bmatrix}$ and integrate the IVP (18a) back for $\mathbf{z}(t)$.
3. Let $\mathbf{y}_{i+1}(t) := \begin{bmatrix} \mathbf{z}(t) \\ R(t)\,\mathbf{z}(t) + (\lambda_{i+1} - \lambda_i)\,\mathbf{r}(t) \end{bmatrix}$.

The Riccati Type 1 method would involve computing

$$f(\lambda) := \det(B_{21} + B_{22}R(1)) \tag{19}$$

(or some other measure of singularity of $B_{21} + B_{22} R(1)$). Note that $f(\lambda) = 0$ iff $Q(\lambda)$ is singular, and that with this approach only (16) need be solved for each update of λ.

The other IV method which we consider is continuous orthonormalization. It has received considerable recent interest, e.g., see [21] for an extensive analysis and [29] for a very readable account of the method. However, it is the only method we consider which has not been extensively compared computationally with other methods in the BVODE setting (the only comparison of which we are aware is [15], and there computer memory limitations play a role). One difficulty with a comparison of either of these IV methods is in deciding from amongst a host of possible implementations. While we have implemented Riccati and continuous orthonormalization following [14] and [12, 23], respectively, any choice is to some extent arbitrary, and other possibilities include those in [5, 21, 22].

Continuous orthonormalization can be adapted as a Type 3 algorithm in a way similar to what we have done for the Riccati method, but for reasons given in Section 4, we have only implemented continuous orthonormalization as a Type 1 algorithm. Since this has been done by Davey [12] and Meyer [23], and since our implementations are very close to theirs, the description here is brief.

Given $T_1(0) \in \mathbf{C}^{n \times q}$ with orthogonal columns and such that

$$B_1 T_1(0) = 0, \tag{20a}$$

we solve an ODE of the form

$$T_1' = A(t, \lambda)\, T_1 + T_1 G, \qquad 0 < t < 1. \tag{20b}$$

If we require $T_1^H T_1' = 0$ (so that $(T_1^H T_1)' \equiv 0$), then by choosing

$$G = -T_1^H A T_1, \tag{21}$$

T_1 spans the same subspace as the columns of the fundamental solution components Y_1 of (20) satisfying $B_1 Y_1(0) = 0$ (which ensures that $Y_1(t)$, and hence $T_1(t)$, contains the fast increasing components). By trying to preserve $T_1^H(t)\, T_1(t) = I$ in the computation, excessive growth of the components is prevented. Interestingly, under the assumption that this orthogonality holds exactly (and some less stringent ones), the IVP (20) is shown to be stable in [21]. Unfortunately, it is not obvious how the computation should be arranged in practice, because substituting (21) directly into (20b) is liable to instability, so in [12, 23],

$$G = -(T_1^H T_1)^{-1}\, T_1^H A T_1 \tag{22}$$

is formed and then this is substituted into (20b). If $T_1^H T_1 = D(I - B)$, where $-DB$ is the (computed) off-diagonal elements, then solving the Orr–Sommerfeld equation Meyer [23] successfully uses the substitution

$$(T_1^H T_1)^{-1} = D^{-1} \tag{23}$$

in (22), although in other applications he uses $(I+B)\,D^{-1}$ [24]. Regardless of how T_1 is computed, the overall goal is to iterate on λ until $\det(B_2 T_1(1)) = 0$. Davey and Meyer describe different methods for computing the eigenfunction $\mathbf{y}^*(t)$ once an eigenvalue λ^* has been determined, and both are used in Section 4.

The global method used here is spline collocation. For this, rather highly developed software is available. We use two descendents of COLSYS, called COLPAR, which is designed to handle parameters directly, and COLCON, which does automatic continuation in a single parameter [6]. Salient features of the codes include being designed for high order ODEs, providing a global spline solution, and having sophisticated mesh selection and nonlinear iteration strategies.

We also use a spectral method, for which the collocation solution is expressed as an expansion of orthogonal functions; the eigenpairs of the discrete linear system approximate the eigenpairs of the original eigenvalue problem. Arguments for the suitability of a Chebyshev expansion for solving the Orr–Sommerfeld equation and a description of the method are given by Orszag [26].

IV. Numerical Results

In this section, we solve the Orr–Sommerfeld equation numerically. This well-known equation, obtained by reduction after linearizing the Navier–Stokes equations, has the form

$$L\phi(t) = \lambda M\phi(t), \qquad 0 < t < 1, \tag{24a}$$

where $L := (-D^2+\alpha^2)^2 + i\alpha R_e[U(-D^2+\alpha) + U'']$, $M := -D^2+\alpha^2$ $(D := d/dt)$, $\lambda = i\alpha R_e c$, and R_e is the Reynolds number. Our problem setup here follows that in [12, 23]. In particular, we consider Poiseuille flow, with $U(t) = 1-t^2$, $\alpha = 1$, and the BCs

$$\phi'(0) = \phi'''(0) = 0, \qquad \phi(1) = \phi'(1) = 0. \tag{24b}$$

Rather than converting to a first-order system using the obvious choice of variables $(\phi, \phi', \phi'', \phi''')^{\mathrm{T}}$, the variables $\mathbf{y} = (\phi, \phi', \phi'' - \alpha^2\phi, \phi''' - \alpha^2\phi')^{\mathrm{T}}$ turn out to be computationally better scaled. This gives

$$\mathbf{y}'(t) = A(t)\,\mathbf{y}, \qquad 0 < t < 1 \tag{25a}$$

$$y_2(0) = y_4(0) = 0, \qquad y_1(1) = y_2(1) = 0, \tag{25b}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \alpha^2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ a & 0 & b & 0 \end{bmatrix}, \qquad a := 2i\alpha R_e, \qquad b := \alpha^2 + i\alpha R_e(1-t^2-c). \tag{26}$$

The eigenfunction changes rapidly near $t = 1$ (e.g., see [12]), and a convenient (and fairly stable) normalization for the eigenfunction turns out to be

$$y_1(0) = \phi(0) = 1. \tag{25c}$$

Thus, if this normalization is used, the IV methods proceed most naturally from $t = 1$ to $t = 0$ for their initial integration, so in the notation of (14b), the point $t = 0$ and BC matrix B_1 correspond to having first implicitly done a change of variables $[0, 1] \to [1, 0]$.

Our goal for this problem is to compute the eigenvalue of smallest mode (i.e., the one with smallest imaginary part, or in some sense the most unstable one). For completeness sake, these eigenvalues are given for various values of R_e in Table I below. These agree with values for $R_e = 10^4$ in [26] and for $R_e = 10^5$ to 10^9 in [12] to the number of digits shown in those papers.

We have found that one of the most useful methods for computing eigenvalues for moderate values of R_e for this problem is the spectral method as in [26]. In particular, using 39 terms in the Chebyshev expansion when $R_e = 10^4$ and 49 terms when $R_e = 10^5$, our spectral method implementation gives approximations accurate to at least eight digits. However, for larger values of R_e the method encounters difficulties, and for $R_e = 10^6$ it gives overflow. (Solving problems with larger R_e by a spectral method would probably require a suitably sophisticated approach which utilizes *local* basis function expansions in order to provide accurate approximation of the eigenfunction near $t = 1$.) In our experience, this spectral method, when successful, always gave a lowest mode approximation which corresponds to the actual one. In general, though, it is possible to be misled by spurious computed modes. This could easily be discovered by refining such approximations with one of the iterative approaches, or spurious modes could be eliminated by using a modified spectral method [16].

TABLE I

Smallest Mode Eigenvalue/Initial Approximations

	Eigenvalue	
R_e	Real	Imaginary
10^4	0.23752649	0.00373967
10^5	0.14592479	−0.01504204
10^6	0.06659252	−0.01398327
10^7	0.03064130	−0.00726049
10^8	0.01417134	−0.00351239
10^9	0.00656630	−0.00166002
10^{10}	0.00304508	−0.00077699
10^{11}	0.00141275	−0.00036208
10^{12}	0.00065558	−0.00016838

The implementation of Approach 1 used here is straightforward. For continuous orthonormalization, our method of integrating (20b), (26) with initial conditions $T_1(1) = \begin{bmatrix} 0 \\ I \end{bmatrix}$ is as described by Davey [12]. For Meyer's modification (23), the change is that G in (22) simplifies to

$$G = -\begin{bmatrix} \dfrac{\tau_1^H A \tau_1}{\tau_1^H \tau_1} & \dfrac{\tau_1^H A \tau_2}{\tau_1^H \tau_1} \\ \dfrac{\tau_2^H A \tau_1}{\tau_2^H \tau_2} & \dfrac{\tau_2^H A \tau_2}{\tau_2^H \tau_2} \end{bmatrix}, \tag{27}$$

where $T_1 = [\tau_1\ \tau_2] \in \mathbf{C}^{4\times 2}$. This improves the efficiency of the eigenvalue approximation by about 50%. For the actual numerical integration, the IVPs are converted to real problems. Unless otherwise stated, the NAG Runge–Kutta–Merson (nonstiff) code D02BBF is used, with a mixed absolute/relative error tolerance. We have found that a multi-step method often performs better, although the difference is such that it does not qualitatively change the conclusions presented here. The stiff check of D02BBF predicts that the IVP for T_1 becomes stiff for $R_e > 10^6$, and the corresponding computation time for D02BBF increases rapidly around this point. However, for our implementation, using the NAG stiff solver D02BBF *does not* qualitatively improve the computation. These results seemingly confirm the observation in [22], and the remark by Davey [12] that these IVPs for continuous orthonormalization are nonstiff, although the matter clearly deserves further analytical study.

After an eigenvalue has been calculated accurately, its corresponding eigenfunction is found. For this, one writes $\mathbf{y}(t) = T_1(t)\,\boldsymbol{\alpha}(t)$ and solves an IVP for $\boldsymbol{\alpha}(t)$ (from $t=0$ to $t=1$ if (25c) is used). Davey integrates for both $T_1(t)$ and $\boldsymbol{\alpha}(t)$. Since the integration for $T_1(t)$ is unstable in this direction, he suggests storing $T_1(t)$ at some interior points during the solution of (20b), and then on the return integration, restart the (still unstable) IVPs for T_1 and $\boldsymbol{\alpha}$ at these points. Meyer [23] also saves $T_1(t)$ values, but after finding $\mathbf{y}(1)$ integrates the *original* (unstable) ODE (1a) from $t=1$ to $t=0$, projecting the computed solution into span $\{T_1(t)\}$ at these restart points. We consider both of these two algorithms. For simplicity of implementation, we use 1000 *equally spaced* interior restart points, though an efficient implementation would choose these points adaptively.

Starting values for λ_0 are four significant digit (rounded) approximations to the exact eigenvalues. Results are not significantly different for the range of values of R_e considered when using less accurate starting values, as long as convergence still takes place—generally this requires one to two digit accuracy in λ_0. The iterative method used to compute the eigenvalue with Approach 1 is a complex secant iteration $\lambda_{i+1} = \lambda_i - f(\lambda_i)/(f(\lambda_i) - f(\lambda_{i-1}))/(\lambda_i - \lambda_{i-1})$, where $f(\lambda) = \det(\bar{B}_2 T_1(0))$ and λ_1 is obtained by adding ± 1 to the fourth significant digit of λ_0.

For the Riccati method, the requirement that B_{12} be nonsingular (where B_1

corresponds to the BCs at $t=1$) necessitates an initial change of variables, and we use $\mathbf{y}=(\phi''-\alpha^2\phi, \phi'''-\alpha^2\phi', \phi, \phi')^{\mathrm{T}}$. As a result, $A=\left[\begin{smallmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{smallmatrix}\right]$ with

$$A_{11}=\begin{bmatrix} 0 & 1 \\ b & 0 \end{bmatrix}, \quad A_{12}=\begin{bmatrix} 0 & 0 \\ a & 0 \end{bmatrix}, \quad A_{21}=\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad A_{22}=\begin{bmatrix} 0 & 1 \\ \alpha^2 & 0 \end{bmatrix}$$

$$B_1=[B_{11}, B_{12}]=\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B_2=[B_{21}, B_{22}]=\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \tag{28}$$

so the Riccati differential equation (16) for $R=\left[\begin{smallmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{smallmatrix}\right]$ becomes $R(1)=0$,

$$\begin{aligned} R'_{11} &= R_{21}-R_{12}(b+aR_{11}) \\ R'_{12} &= R_{22}-R_{11}-aR_{12}^2 \\ R'_{21} &= 1+\alpha^2 R_{11}-R_{22}(b+aR_{11}) \\ R'_{22} &= R_{12}(\alpha^2-aR_{22})-R_{21} \end{aligned} \tag{29}$$

and from $B_{21}+B_{22}R(0)=\left[\begin{smallmatrix} R_{21}(0) & R_{22}(0) \\ 0 & 1 \end{smallmatrix}\right]$, the desired terminal condition is just

$$f(\lambda)=\det(B_{21}+B_{22}R(0))=R_{21}(0)=0. \tag{30}$$

Again, the system is converted to real IVPs, standard software is used, and complex secant iteration in λ used to satisfy (30). Results mainly using the nonstiff solver D02BBF are given in Table II. For large R_e, (29) is stiff [13], as is clearly demonstrated by the fact that D02EBF becomes much more efficient. Corresponding to the last column of Table II, the Riccati method successfully solves the eigenvalue problems for $R_e=10^{10}$, 10^{11}, and 10^{12} with normalized times as described below of 6.48, 9.10, and 9.13, respectively.

After an accurate eigenvalue λ^* has been computed, the corresponding eigenfunction $\mathbf{y}^*$ can be found by solving $[B_{21}+B_{22}R(0)]\,\mathbf{z}(0)=0$ for a nonzero vector

TABLE II

Eigenvalue Calculations—Approach 1

	Continuous orthonormalization		Riccati method		
R_e	Davey	Meyer	From $t=1$	From $t=0$	From $t=0$ (stiff solver)
10^4	10.22	6.09	2.58	2.87	4.21
10^5	14.63	8.32	4.32	1.62	6.96
10^6	40.78	11.29	5.46	6.83	7.57
10^7	67.52	38.15	8.72	11.67	8.68
10^8	125	40.52	17.94	20.88	9.47
10^9	257.46	125.68	41.13	24.81	8.27

satisfying the normalization $z_1(0) = 1$, and integrating the ODE $\mathbf{z}' = [A_{11} + A_{12}R]\,\mathbf{z}$ from $t = 0$ to $t = 1$. Then $\mathbf{y}^* = \begin{bmatrix} \mathbf{z} \\ R\mathbf{z} \end{bmatrix}$. From (25a), (28), the IVP for $\mathbf{z}$ is

$$z_1(0) = 1, \qquad z_2(0) = 0, \tag{31a}$$

$$z_1' = z_2 \tag{31b}$$

$$z_2' = (b + aR_{11})\, z_1 + aR_{12} z_2.$$

For this return integration, we use a very crude implementation, roughly like that for continuous orthonormalization, viz., R_{11} and R_{12} and their derivatives obtained from (29) are stored at 1000 equally spaced interior points, and their cubic Hermite interpolants are then used when integrating (31b). For large Reynolds number, storing values of $(b + aR_{11})$ and interpolating them directly was found to be less prone to cancellation errors. For moderate values of R_e, computer times are not markedly less than those for the continuous orthonormalization eigenfunction computations (though roughly 50% less than with Meyer's method). For larger R_e, adaptively choosing interior points to store R would improve matters significantly, since the solution to the stiffly stable IVP for R tends to exhibit layers only where the eigenfunction solutions do (e.g., see [14]).

A summary of the numerical results for these IV methods using Approach 1 is given in Tables II and III. Tolerances in the IV codes were chosen such that roughly eight digit accuracy is obtained for the final eigenvalue and eigenfunctions. This requires a somewhat smaller tolerance for continuous orthonormalization than for Riccati, at least with the codes used. For example, a tolerance of 10^{-9} is typically required with continuous orthonormalization, but often only 10^{-6} for Riccati. This is of course very code dependent, but slight changes in these tolerances do not qualitatively change the results. The numerical tests have been carried out on various machines (including runs on PCs with all of the methods), and those reported here are for double precision computations on an IBM 3081 and DEC 8650; unit roundoff for both is around 10^{-15}. The computer times have been normalized to correspond to the IBM CPU time in seconds (where the normalization factor is

TABLE III

Eigenfunction Calculations—Approach 1

	Continuous orthonormalization		
R_e	Davey	Meyer	Riccati from $t = 1$, nonstiff solver
10^4	12.93	7.61	1.98
10^5	17.19	11.13	4.67
10^6	23.94	14.84	7.23
10^7	41.60	25.12	11.76
10^8	96.06	48.90	21.62
10^9	Failed	Failed	51.52

found by making identical runs on both computers). Of course, what is important here is the relative difference between the methods, as the implementations are far from polished but of roughly equal sophistication.

All in all, the Riccati implementation is more efficient than continuous orthonormalization, particularly for large Reynolds numbers. Moreover, convergence of Riccati is generally found to be less sensitive to the accuracy of the initial guess. In Table IV, the number of iterations and normalized CPU times needed for convergence of the iteration corresponding to the right hand column of Table II are shown. Riccati with Approach 1 is denoted by RICC1, and (i) and (ii) correspond to using one and two significant digits, respectively, for the initial eigenvalue approximations λ_0 and λ_1. Note how computer time increases slowly for increasing R_e, using the stiff integrator D02EBF with a mixed error tolerance.

We have not implemented any of the IV methods using Approach 2. In [15], trouble is reported with continuous orthonormalization due to degenerating accuracy when computing (orthogonal) $T_1(t)$ and using (1c), but we have not substantiated this ourselves. On the basis of its superior performance here and in [22], we have modified the Riccati method, as described in Section 3, to be used with Approach 3. Recall that now an eigenvalue and an eigenvector approximation are computed at each iteration. To begin the iteration given only λ_0, the initial eigenfunction approximation $\mathbf{y}_0$ is computed by solving (29) for R and (31) for $\mathbf{z}$; then $\mathbf{y}_0 = \begin{bmatrix} \mathbf{z} \\ R\mathbf{z} \end{bmatrix}$. As we expected, it is found to be more robust than Approach 1, at least for moderate R_e. Table IV shows the number of iterations required for convergence for R_e between 10^4 and 10^7, with RICC3 denoting this Approach 3 implementation and (i) and (ii) corresponding to initial guesses for λ_0 with one and two significant digits, respectively. Computer times are not given, being unduly affected by the crude implementation of the eigenfunction computation. In fact, results for $R_e > 10^7$ are more erratic and probably unreliable as a test of the approach, apparently

TABLE IV

Riccati Eigenvalue Results for Various Initial Guesses

	RICC1 (i)		RICC1 (ii)		RICC3 iterations	
R_e	Time	Iterations	Time	Iterations	(i)	(ii)
10^4	Diverges		6.28	6	5	4
10^5	16.33	15	8.17	7	8	5
10^6	24.99	17	8.37	6	7	5
10^7	16.26	10	9.35	6	5	5
10^8	Diverges		10.02	6		
10^9	22.55	12	11.18	6		
10^{10}	11.44	6	12.79	6		
10^{11}	Diverges		12.76	6		
10^{12}	16.59	7	12.03	5		

because the implementation prevents obtaining sufficiently accurate eigenfunction approximations, although this requires more study.

The spline collocation code COLPAR is used with both Approaches 2 and 3 to solve the Orr–Sommerfeld equation. Because of its ability to handle high order ODEs, the real and imaginary parts of (24a), (24b) are computed directly using two fourth-order ODEs. For Approach 2, two trivial first-order ODEs and BCs corresponding to the complex ODE $\lambda' = 0$ and BC $\phi(0) = 1$ are added. For Approach 3, the BC $\phi(0) = 1$ is added to (24a), (24b), and COLPAR treats λ directly as a parameter.

COLPAR is essentially the fixed parameter version of the code COLCON [6], which does continuation in a single parameter. COLCON uses a Gauss–Newton iteration in solving the discretization equations for the collocation solution coefficients $\mathbf{z}_i$ and parameter values $\boldsymbol{\alpha}_i$. The linear systems have the form

$$D\mathbf{F}(\mathbf{z}_i, \boldsymbol{\alpha}_i)\begin{bmatrix} \delta\mathbf{z}_i \\ \delta\boldsymbol{\alpha}_i \end{bmatrix} = -\mathbf{F}(\mathbf{z}_i, \boldsymbol{\alpha}_i) \tag{32}$$

(cf. [6, Eq. (3.20)]) and have rank deficiency the dimension of $\boldsymbol{\alpha}_i$. The solution strategy involves *implicitly* adjoining extra equations to (32) such that $\left[\begin{smallmatrix} \delta\mathbf{z}_i \\ \delta\boldsymbol{\alpha}_i \end{smallmatrix}\right]$ is made orthogonal to the null space of $D\mathbf{F}(\mathbf{z}_i, \boldsymbol{\alpha}_i)$, and some theoretical justification for this choice is given. Not surprisingly, this approach can be interpreted in our setting in a natural way. All except the last equation of (11′) can be viewed as a condensed set of discretization equations (e.g., for collocation, as a condensation of (32)). For the block bidiagonal matrices of the type produced and solved by COLCON, full column pivoting in the last block insures that the parameters $\boldsymbol{\alpha}_i$ come into play so that one does not try to invert, in the notation of (11′), $Q(\lambda)$. Obviously, when a reliable explicit normalization (like (25c) appears to be) is unavailable, as would in general be the case, then a more careful choice like this used in COLCON is essential.

For Approaches 2 and 3 we need an initial estimate λ_0 for the eigenvalue and $\phi_0(t)$ for the corresponding eigenfunction, and the approximations need to be accurate for the computed eigenvalue to be the desired one. One way to obtain such an accurate approximation is to use COLPAR with λ_0 treated as a known constant and solve the standard linear BVODE (24a), (24b), with the BC $\phi'''(0) = 0$ replaced by $\phi(0) = 1$, for $\phi_0(t)$. Thus the eigenvalue estimate is held constant and a reasonable approximation for the eigenfunction is obtained. If a tolerance TOL is desired on the final eigenvalue/eigenfunction (λ^*, ϕ^*), then the requested tolerance for ϕ_0 is $\sqrt{\text{TOL}}$. Unless stated otherwise, a relative/absolute tolerance of 10^{-8} is used for the components ϕ and λ, the initial mesh is the default one of five equal spaced subintervals, and the spline basis with four collocation points per subinterval is used.

The numerical results for COLPAR with Approach 2 are not markedly different than those with Approach 3, except that the former takes more computer time and storage, due to the larger ODE system. Table V shows, for COLPAR with

TABLE V

Spline Collocation Results

	COLPAR	COLCON	
R_e	Time	Time	Steps
10^4	2.30		
10^5	3.80	87.34	14
10^6	3.39	191.2	19
10^7	4.07	288.5	24
10^8	4.93	388.6	24
10^9	4.57	369.2	29
10^{10}	5.42		
10^{11}	5.57		
10^{12}	6.83		

Approach 2, the corresponding results to those in Tables II and III for the IV methods. Comparable results have been obtained with Approach 3, which is typically about 20% faster. The efficiency of the collocation method is apparent, especially since the computed solution, which increases slowly with R_e, involves both the eigenvalue and eigenfunction computation.

COLPAR is also extremely robust and provides reliable error estimates for the computed eigenpair. It converges to the smallest mode eigenvalue for $R_e = 10^4, ..., 10^9$ using an initial eigenvalue approximation with at least one significant digit. Given only the correct order of the smallest mode eigenvalue, it converges to it somewhat more reliably than RICC3, although the latter frequently still converges (but to another eigenvalue), making comparison slightly more delicate. While computer times are a measure of the efficiency of the codes, the considerable imbalance between the level of sophistication of COLPAR and the implementations of the IV methods makes comparison between these two types of methods problematic. It is exacerbated further because the IV methods choose a new mesh each (secant or Newton) iteration, while COLPAR uses a damped Newton iteration on a given mesh, and then adapts the mesh if necessary in an attempt to equidistribute the error.

With this in mind, the iteration process for COLPAR for $R_e = 10^4$ and 10^7, with λ_0 having two significant digits accuracy, is examined. We give (a) the number of mesh points used during the calculation of ϕ_0, followed by (b) the number of points in the mesh sequence chosen during the Newton iteration on (λ_i, ϕ_i):

$R_e = 10^4$. (a) 5, 10, 20; (b) 20, 16, 32

$R_e = 10^7$. (a) 5, 10, 20, 16, 32, 16, 32, 16, 32, 64; (b) 64, 35, 70.

Only one Newton iteration, perhaps with quasi-Newton iteration involving a fixed Jacobian, is done on all of the meshes except the first one in (b), where three and four iterations are done for $R_e = 10^4$ and 10^7, respectively. Normalized times are

6.35 and 19.97 s, respectively. In general, since an accurate approximation to an eigenvalue may be unavailable, we may need to compute (say) the smallest mode eigenvalue for a range of values of R_e (or a bifurcation analysis may be required). Then, instead of computing (λ^*, ϕ^*) directly as above, it becomes necessary to use continuation. COLPAR is essentially the fixed parameter version of the code COLCON [6], which does continuation in a single parameter.

Using the Reynolds number as the continuation parameter, we have solved (24) with COLCON for $10^4 \leqslant R_e \leqslant 10^{10}$, and a selection of these results are shown in Table V. The computer times correspond to independent runs solving for $R_e = 10^p \to 10^{p+1}$, using the (automatically selected) number of continuation steps shown. COLCON is robust and reliable for this type of problem, although alternative codes are available. For example, Ache [2] recently used the BVP code PASVAR and continuation to solve similar eigenvalue problems using Approach 2. Note from Table V that, despite the powerful adaptive mesh strategy and sophisticated technique for choosing the continuation step for COLCON, the computer time initially increases significantly with R_e and is fairly large as compared to that for COLPAR. This is because small changes in R_e can still cause significant movement in the layer regions of the eigenfunction, so in addition to taking cautious initial continuation steps, relatively small steps are mandatory throughout. Also, keeping the number of mesh points (a major indicator of computer time) to a minimum is more difficult than when only solving for a single parameter value, as COLPAR does.

N.B. For this particular problem, accurate approximations to λ^* are readily obtained for large values of the Reynolds number because both $\mathrm{Re}(R_e)$ and $\mathrm{Im}(R_e)$ essentially vary linearly with $\log(R_e)$.

V. Conclusions

The interrelationships between three natural approaches for solving the eigenvalue problem (1a), (1b) have been shown. The technique of Keller, which involves enlarging the problem as a standard BVODE, has been interpreted in terms of a simplified strategy explicitly dealing with the eigenvalue λ as a parameter (cf. Approach 3). We have considered, for various approaches, a spline collocation method and the Riccati and continuous orthonormalization methods. With these three methods, we have computed the eigenpair of lowest mode for the Orr–Sommerfeld equation for a wide range of values of the Reynolds number R_e. The collocation method, having the generally most robust implementation, has been successfully used with relatively inaccurate initial approximations, while typically the IV methods needed more accurate approximations. The spectral method cheaply provides such initial approximations for small and moderate values of R_e.

The Riccati method performs best of the two IV methods. This is not to say that implementations of continuous orthonormalization cannot be made still more competitive, only that stability and practical implementation questions remain and that

the performance of our implementation of it was inferior to the corresponding one for the Riccati method. The potential for stiffness of the IVPs using continuous orthonormalization needs to be better understood, as the nonlinear equation (20b) appears considerably more difficult to handle for stiff methods like BDF than the quadratic nonlinearity of the Riccati method. While no reimbedding was necessary for the Riccati method for the Orr–Sommerfeld equation, for most problems a reimbedding strategy does not qualitatively affect the relative performance (see also [14]).

Conclusions about the methods considered here are far from being definitive, being of course based upon computations for the Orr–Sommerfeld equation. The implementations we use appear competitive with those used elsewhere for solving problems of the form (1a), (1b), as to our knowledge the Orr–Sommerfeld problem has been solved here for larger Reynolds numbers than has been published heretofore. While the computations for extremely large R_e are interesting in themselves, a major use is to determine the relative robustness of the methods as a predictive measure of their utility in a variety of contexts. Navier–Stokes computations for large R_e are becoming increasingly important (e.g., see [10] and, in a different setting, [25]). We feel that the numerics here are indicative of the general picture as far as how the methods would perform for most difficult eigenvalue problems.

Our comparison of the spline collocation and Riccati *methods* is at best rudimentary, but comparison of their *implementations* is not—COLPAR wins hands down. It is highly sophisticated software using robust nonlinear iteration and mesh selection strategies. For the Riccati implementation, we make use of the special form and properties of the Orr–Sommerfeld problem (for example, in the normalization and choice of variables, and in the explicit reduction of number of components of R needed), but the nonlinear iteration, mesh selection, and interpolation are extremely crude. These factors only partially compensate for each other, but in sum there is reason to hope that a competitive code based upon the Riccati method is feasible and should be developed. Nevertheless, consideration should be given to the presumed advantage of collocation in handling a high-order problem like (24) directly, as the problem of scaling variables when converting high-order ODEs to first-order systems can need special care for any given problem. Questions such as the choice of nonlinear iteration scheme and the extra normalizing BC for the particular methods would need to be examined. Finally, the computational experience with IV methods has emphasized the need for having IV codes with the facility for providing global error estimates and continuous solution profiles, aspects which have received recent, though somewhat belated, attention by IV code developers.

One useful purpose of a robust implementation of an IV method like a Riccati method would be to supplement a collocation code like COLPAR. Reproducing numerical results with an entirely different code and method is a time-honoured way of assuring reasonable reliability of numerical results. But while it is undoubtedly worthwhile to investigate other IV and global methods further, it is also manifestly clear that COLPAR has the capacity to solve very difficult problems.

Many of its adaptive features have been intensively developed and extensively tested on a wide assortment of problems. Its robustness should be better known, and more general use made of such software, particularly by the scientists and engineers whose interest is in the reliable solution to physical problems and not the computation itself.

REFERENCES

1. A. A. ABRAMOV, *U.S.S.R. Comput. Math. and Math. Phys.* **1**, 617 (1961).
2. G. A. ACHE, *SIAM J. Sci. Statist. Comput.* **10**, 1097 (1989).
3. U. ASCHER, *J. Comput. Phys.* **34**, 401 (1980).
4. U. M. ASCHER, R. M. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1988), p. 135.
5. I. BABUSKA AND V. MAJER, *SIAM J. Numer. Anal.* **24**, 1301 (1987).
6. G. BADER AND P. KUNKEL, *SIAM J. Sci. Statist. Comput.* **10**, 72 (1989).
7. P. B. BAILEY, M. K. GORDON, AND L. F. SHAMPINE, *ACM Trans. Math. Software* **4**, 193 (1978).
8. N. S. BAKHVALOV, *Numerical Methods* (MIR, Moscow, 1977, Russian edition 1975), p. 601.
9. C. DE BOOR AND B. SWARTZ, *Math. Comput.* **35**, 679 (1980).
10. W. W. BOWER, J. T. KEGELMAN, A. PAL, AND G. MEYER, *Phys. Fluids* **30**, 998 (1987).
11. J. S. BRAMLEY AND S. C. R. DENNIS, *J. Math. Anal. Appl.* **101**, 30 (1984).
12. A. DAVEY, *J. Comput. Phys.* **51**, 343 (1983).
13. L. DIECI, M. R. OSBORNE, AND R. D. RUSSELL, *SIAM J. Numer. Anal.* **25**, 1055 (1988).
14. L. DIECI, M. R. OSBORNE, AND R. D. RUSSELL, *SIAM J. Numer. Anal.* **25**, 1074 (1988).
15. M. EIDENSCHINK, MSc. thesis, Georgia Institute of Technology, 1988 (unpublished).
16. D. R. GARDNER, S. A. TROGDON, AND R. W. DOUGLASS, *J. Comput. Phys.* **80**, 137 (1989).
17. I. H. HERRON, *SIAM Rev.* **29**, 597 (1987).
18. H. B. KELLER, *Numerical Solution of Two Point Boundary Value Problems* (SIAM, Philadelphia, 1976), p. 45.
19. M. LENTINI, M. R. OSBORNE, AND R. D. RUSSELL, *SIAM J. Numer. Anal.* **22**, 280 (1985).
20. M. LENTINI AND V. PEREYRA, *Mat. Apl. Comput.* **2** (1983).
21. P. M. VAN LOON, *Continuous Decoupling Transformations for Linear Boundary Value Problems* (Math. Centrum, Amsterdam, 1988), p. 45.
22. P. M. VAN LOON AND R. M. M. MATTHEIJ, *Austral. Math. Soc. Ser. B* **29**, 282 (1988).
23. G. H. MEYER, *J. Comput. Phys.* **62**, 248 (1986).
24. G. H. MEYER, School of Math., Georgia Inst. of Tech., Atlanta, GA, private communication (1989).
25. B. MULLER AND A. RIZZI, *Int. J. Numer. Methods Fluids* **9**, 943 (1989).
26. S. A. ORSZAG, *J. Fluid Mech.* **50**, 689 (1971).
27. M. R. SCOTT AND H. A. WATTS, *SIAM J. Numer. Anal.* **14**, 40 (1977).
28. A. G. SLEPTSOV, "The Spline-Collocation and the Spline-Galerkin Methods for Orr–Sommerfeld Problem," in *Numerical Boundary Value ODEs*, edited by U. Ascher and R. D. Russell (Birkhauser, Boston. 1985), p. 137.
29. K. WRIGHT, Univ. Newcastle upon Tyne Computing Lab Tech. Rep. 257, 1988 (unpublished).