JOURNAL OF COMPUTATIONAL PHYSICS 5, 169–187 (1970)

A Matrix Method for Ordinary Differential Eigenvalue Problems

JOHN GARY AND RICHARD HELGASON

National Center for Atmospheric Research, Boulder, Colorado 80302

Received March 24, 1969

This paper describes a method for solving ordinary differential eigenvalue problems of the form $N(u) + \lambda M(u) = 0$, where N and M are linear differential operators and $u(x)$ is a scaler variable. The boundary conditions are independent of λ. The problem is transformed into a matrix problem $|A + \lambda B| = 0$. This is reduced to the standard eigenvalue problem $|\hat{A} + \lambda I| = 0$ which is then solved by the $Q - R$ algorithm. The computer program is organized so that it can solve a wide range of problems with minimal effort on the user's part. The method is applied to a hydrodynamic stability problem and compared to the shooting method.

1. INTRODUCTION

We are concerned with the determination of eigenvalues of ordinary differential equations. An example is the harmonic equation

$$u^{(2)}(x) + \lambda u(x) = 0,$$
$$u(0) = u(1) = 0,$$

whose solutions are

$$\lambda = n^2\pi^2,$$
$$u(x) = \sin \pi n x,$$
$$n = 1, 2, 3, \ldots .$$

Another example is the Orr-Sommerfeld equation which arises in the study of the stability of viscous fluid flow. It is this problem which motivated our development of the method. One way to solve these problems is the "shooting method." One way to apply the shooting method to the above harmonic equation is the following. Assume a value for $u^{(1)}(0)$, say $u^{(1)}(0) = 1$. Assume an initial guess for λ, say λ_0. Using an integration scheme, perhaps Runge-Kutta, integrate the differential equation from $x = 0$ to $x = 1$. Then $u(1)$ is a function of λ, say $u(1, \lambda) = F(\lambda)$. The objective is to choose λ in such a way that $F(\lambda) = 0$. This requires use of a rootfinder such as that of Aitken, Muller, Newton, or Laguerre. This use of a

581/5/2-1

rootfinder is one of the difficulties with the shooting methods. It is difficult to find a rootfinder which will converge regardless of the initial guess. In the hydrodynamic stability problem, we wish to find the least stable mode—a mode being defined by an eigenvalue. If we find just one eigenvalue, we cannot be sure we have the least stable mode, and, thus, we may want the rootfinder to locate many eigenvalues. Many rootfinders will fail to converge after a few eigenvalues have been located.

A second method for these eigenvalue problems is the "matrix method," frequently called the "finite difference method." To solve the harmonic equation above we could define a mesh of points by:

$$x_j = jh, \qquad 0 \leqslant j \leqslant J,$$
$$h = 1/J.$$

Using the obvious finite difference approximation for the second derivative, the equation reduces to a matrix eigenvalue problem [$u_j = u(x_j)$].

$$\begin{vmatrix} -2 & 1 & 0 & \cdots & & \\ 1 & -2 & 1 & 0 & \cdots & \\ 0 & 1 & -2 & 1 & 0 & \cdots \\ \vdots & & & & & \\ 0 & & \cdots & & 1 & -2 & 1 \\ 0 & & \cdots & & 1 & 1 & -2 \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \\ \\ \vdots \\ u_{J-1} \end{vmatrix} + h^2\lambda \begin{vmatrix} u_1 \\ \\ \vdots \\ \\ u_{J-1} \end{vmatrix} = 0$$

or $(A - \mu I)\, u = 0$, with $\mu = -h^2\lambda$.

The primary advantage of the matrix method is the existence of a very reliable method for solving the standard eigenvalue problem $| A - \mu I | = 0$; namely, the *Q-R* algorithm [11]. Using this method, we obtain all the eigenvalues of the matrix. However, the order of the matrix and, thus, the accuracy of the result are restricted by computer speed and memory size. For this reason, the matrix method is poor if applied to a system of equations rather than a single higher order differential equation. A mesh which contains only 25 points applied to a system of 4 second-order equations yields a matrix of order 100. The matrix method will probably give more accurate results on a single eighth-order equation.

Since the computation time for the *Q-R* algorithm increases as the cube of the order of the matrix, it is important to use accurate difference schemes in order to reduce the number of mesh points. This has frequently been done by a "Numerov" type of correction which does not increase the number of mesh points required to approximate a derivative [5]. With some matrix methods this is important, since the matrix which results is "banded", and the method is faster if the bandwidth is minimized [5, 10]. In our example for the harmonic equation, the bandwidth is one; the matrix is tridiagonal. Osborne uses a five-point scheme on the Orr-

Sommerfeld equations which results in a bandwidth of two. Unfortunately, the Q-R algorithm is no faster for a banded matrix than for a full matrix. (It also requires working storage equal to the full matrix even though the matrix is banded.) However, this disadvantage permits us to increase the accuracy by simply adding more points to the difference approximation. With respect to the Q-R computations, it costs us nothing to use a nine-point approximation rather than a five-point approximation.

Many eigenvalue problems have the form $N(u) + \lambda M(u) = 0$, where N is an n-th order differential operator and M is of order m ($m < n$). An example might be

$$u^{(4)}(x) + u^{(2)}(x) + u(x) + \lambda[u^{(2)}(x) + u(x)] = 0.$$

If we replace this equation by a finite difference approximation, we obtain a matrix eigenvalue problem of the form $| A + \lambda B | = 0$. The Q-R algorithm applies only to $| A + \lambda I | = 0$. If B is nonsingular, we can reduce the first problem to

$$| B^{-1}A + \lambda I | = 0.$$

The finite difference approximation we have used invariably produces a singular B matrix. We have developed two methods to handle this singular case. These methods will be described later.

Stengl and Isaacs consider the reduction of the general eigenvalue problem $| A + \lambda B | = 0$ to the standard problem which they then solve by the Q-R method. They orient their program toward oscillatory systems rather than two-point boundary value problems. They seem to require that the matrix B be nonsingular since they state the following on p. 22: "The matrix could also be singular if a row or column was equal to a linear combination of the remaining rows or columns. This would generally indicate the need for a new formulation of the problem." They do not describe a systematic method for such a reformulation.

There are several papers based on the use of a rootfinder to evaluate the zeros of the determint $f(\lambda) = | A + \lambda B |$ [10, 9, 3]. Such methods can be very effective. We feel that a method based on the Q-R algorithm might be, in general, more reliable than one based on a rootfinder. A Q-R method might be good for backup and checking, even if it is slower than other methods.

With the exception of the transformation from the form $| A + \lambda B | = 0$ to $| A + \lambda I | = 0$, the methods we have used are rather standard. The novelty in our approach lies in the combination of these methods to form a computer program which is flexible, easy to apply, and fairly reliable. The method applies to a single differential equation of the form

$$N(u) + \lambda M(u) = 0, \tag{1}$$

$$N(u) = h_n(x)\, u^{(n)}(x) + h_{n-1}(x)\, u^{(n-1)}(x) + \cdots + h_0(x)\, u(x),$$

$$M(u) = g_m(x)\, u^{(m)}(x) + g_{m-1}(x)\, u^{(m-1)}(x) + \cdots + g_0(x)\, u(x).$$

The interval is $a \leqslant x \leqslant b$, and the boundary conditions are

$$\begin{aligned}
&\alpha_{1s}u^{(s)}(a) + \cdots + \alpha_{10}u(a) = 0 \\
&\vdots \\
&\alpha_{ps}u^{(s)}(a) + \cdots + \alpha_{p0}u(a) = 0, \\
&\beta_{1s}u^{(s)}(b) + \cdots + \beta_{10}u(b) = 0, \\
&\vdots \\
&\beta_{rs}u^{(s)}(b) + \cdots + \beta_{r0}u(b) = 0,
\end{aligned} \tag{2}$$

where

$$p + r = n \qquad \text{and} \qquad s < n.$$

Note that the eigenvalue λ is not allowed to appear in the boundary conditions. The programmer is permitted many input options. The program allows a variable mesh; the mesh points need not be equally spaced. "Fictitious" mesh points may be introduced. We can add p_a points to the left of $x = a(0 \leqslant p_a \leqslant p)$ and r_b to the right of $x = b(0 \leqslant r_b \leqslant r)$. The number of points used in the difference approximation of the derivatives is arbitrary (except that it must be odd). The program is written in FORTRAN for the Control Data 6600 computer. We make use of the Control Data microfilm plotter (*dd*80) in this program to produce a graphical representation of the eigenfunction.

The user is required to furnish function subprograms for calculation of the coefficient functions, the mesh transformation, and a subroutine for selecting eigenvalues of interest to the user. He is, thus, insulated from the program structure in that he need only know the calling sequence for each routine he supplies, and the name of a common block used for parameters he supplies in the data. A type of symbolic data input has been implemented which allows data to be input by reference to a data key word. The data associated with the data key is entered in a field-free format. Also, the data structure is such that multiple cases may be run at one time without unnecessary repetition of previously input data.

In the next few sections, we will describe our matrix method. In the main, our method merely amounts to collecting standard procedures and placing them in a single program. The only exception to this is the way we reduce the general eigenvalue problem $|A + \lambda B| = 0$ to the standard $|A + \lambda I| = 0$. We will describe the application of the method to a hydrodynamic stability problem—the Orr-Sommerfeld equation for Poiseuille flow between flat plates. For comparison, we will describe application of the shooting method to the same problem. We did not compare this method with that of Osborne [5], although the comparison would be of considerable interest. Osborne's method uses a banded matrix. It should be much faster than our method. Since he uses Newton's method as a rootfinder, it might be less reliable in cases where one does not have a good initial guess for the eigenvalue.

2. The Finite Difference Approximation

In this section we will describe the transition from the differential equation to the matrix eigenvalue problem $|A + \lambda B| = 0$. We first specify the choice of the mesh points $\{x_j\}$. In some cases, a variable mesh spacing is desirable. In the case of Poiseuille flow between flat plates described later, we need a finer mesh near the boundary point at $x = -1$ (at the plate) than at $x = 0$ (at the center of the channel). We allow the user of the routine to specify a function $s(x)$ which determines the mesh spacing. An equally spaced mesh is used in s. The values for the mesh points x_j are found by solving the equation $s_j = s(x_j)$ for x_j. The user must feed in the functions $s(x)$ and $s^{(1)}(x)$, then the program uses Newton's method to determine the x_j from the s_j.

In some cases, the user may wish to add "fictitious" mesh points outside the end points of the interval $a \leqslant x \leqslant b$. For example, if we wished to solve the problem

$$u^{(2)}(x) + \lambda u(x) = 0,$$
$$u^{(1)}(0) = u^{(1)}(1) = 0,$$

we might use an equally spaced mesh in x, but add a point at $x = -\Delta x$. This would permit use of a centered difference approximation for the boundary condition. If we use a 3-point difference scheme, the matrices A and B would, thus, be

$$A = \begin{vmatrix} 1 & 0 & -1 & 0 & & & \cdots & & 0 \\ 1 & -2 & 1 & 0 & & & & & \\ 0 & 1 & -2 & 1 & 0 & & & & \vdots \\ & & & & \vdots & & & & \\ \vdots & & & & 0 & 1 & -2 & 1 & 0 \\ & & & & & & 1 & -2 & 1 \\ 0 & & & \cdots & & & 1 & 0 & -1 \end{vmatrix}$$

$$B = \begin{vmatrix} 0 & 0 & 0 & & & \cdots & & & 0 \\ 0 & h^2 & 0 & & & & & & \\ 0 & 0 & h^2 & 0 & 0 & & & & \vdots \\ & & & & \vdots & & & & \\ \vdots & & & & 0 & 0 & h^2 & 0 & 0 \\ & & & & & & 0 & h^2 & 0 \\ 0 & & & \cdots & & & 0 & 0 & 0 \end{vmatrix}.$$

The top and bottom rows are obtained from the boundary condition; the remaining rows are obtained from the differential equation. If $\Delta x = 1/(J - 3)$, then the order of the matrices is J. Note that B is singular. If we did not add any points outside the interval, we could use an uncentered difference approximation for the boundary conditions which would again form the first and last rows of the matrices.

The remaining rows would be obtained from the differential equation, except the difference approximation for the second row would now be centered at $x = \Delta x$ instead of $x = 0$. In this case, $\Delta x = 1/(J - 1)$, the order of the matrices is J, and they are given below.

$$A = \begin{vmatrix} -3 & 4 & -1 & 0 & & \cdots & & 0 \\ 1 & -2 & 1 & 0 & & & & \\ 0 & 1 & -2 & 1 & 0 & & & \vdots \\ & & & & \vdots & & & \\ \vdots & & & 0 & 1 & -2 & 1 & 0 \\ & & & & 0 & 1 & -2 & 1 \\ 0 & & \cdots & & 0 & 1 & -4 & 3 \end{vmatrix} \qquad B = \begin{vmatrix} 0 & 0 & & \cdots & & 0 \\ 0 & h^2 & 0 & & & \\ \vdots & & & & & \vdots \\ & & & 0 & h^2 & 0 \\ 0 & \cdots & & 0 & 0 & 0 \end{vmatrix}.$$

We always use $2J + 1$ mesh points in these finite difference approximations of the derivatives. In the examples above, $J = 1$. We always use centered difference approximations, if possible; otherwise, one-sided difference approximations. We have a subroutine which computes the Lagrangian interpolation coefficients (in double precision) [4]. The variable mesh spacing and the one-sided difference formulas are, of course, no problem. To produce the j-th row in the matrix A, we take the Lagrangian interpolation coefficients for $u^{(\nu)}$, evaluated at x_k, and multiply by $h_\nu(x_k)$, where k is a function of j [see Eqs. (1)]. We do the same for all the terms $h_\nu u^{(\nu)}$ in the differential operator $N(u)$ and add to obtain the elements in the matrix A. The matrix B is obtained from the differential operator $M(u)$ in the same manner.

If there are p boundary conditions at the left boundary point, then the first p rows of A are always obtained from these boundary conditions. The first p rows of B always vanish. The Lagrangian interpolation coefficients for the p boundary conditions are evaluated at $x = a$ since the boundary conditions are set at $x = a$. These interpolation formulas use the points $x_1, \ldots, x_{2J+1}$. The points $x_1, \ldots, x_{p_a}$ are located to the left of $x = a$. (We may have $p_a = 0$, in which case $x_1 = a$.) The $(p + 1)$-th row of the matrix A is a finite difference approximation to the differential operator $N(u)$, evaluated at the point x_{p+1}. Note that $x_{p+1} \geqslant a$. The $(p + 2)$-th row is an approximation to $N(u)$ evaluated at x_{p+2}, and similarly until the other boundary is reached.

The method we have just described results in a singular B matrix, but this method is general and, thus, easy to automate on the computer. We describe methods to deal with the singular B matrix in the next section.

3. Reduction to the Standard Eigenvalue Problem

Here we describe two methods for reducing our general eigenvalue problem $|A + \lambda B| = 0$ to a standard eigenvalue problem $|\hat{A} + \lambda I| = 0$ which has the

same eigenvalues. The first method (special reduction) is much faster than the second (general reduction) but the special reduction is more likely to fail. The program uses the special reduction first, then if this fails, it trys the general reduction.

Our matrices A and B have the form shown in Figs. 1 and 2 (remember that p is the number of boundary conditions at $x = a$ and r the number at $x = b$, and we let $J_2 = 2J + 1$ be the number of points used in the finite difference approximation).

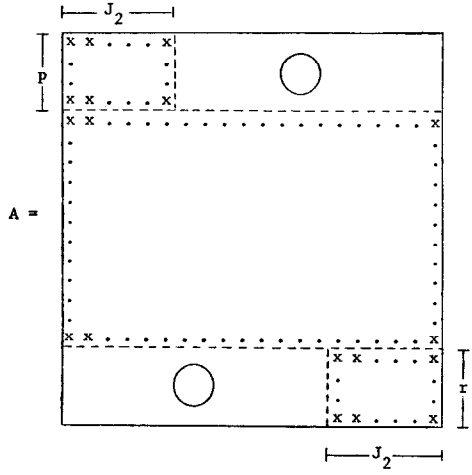

FIG. 1. The A matrix before reduction.

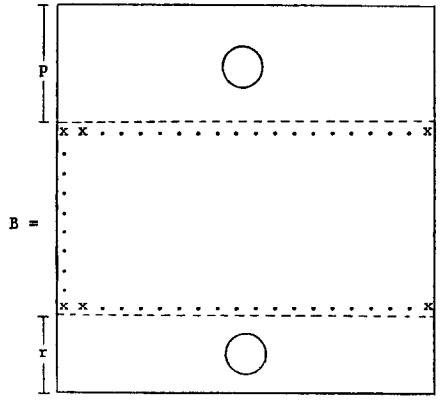

FIG. 2. The B matrix before reduction.

We now use a sequence of elementary column transformations to define transformed matrices $\tilde{A}$ and $\tilde{B}$ (the product of the elementary column transformations we denote by Q).

$$\tilde{A} = AQ, \qquad \tilde{B} = BQ.$$

Note that the eigenvalues of $|\tilde{A} + \lambda\tilde{B}| = 0$ are the same as those for $|A + \lambda B| = 0$. We define the transformations as follows.

Choose a reasonably large element (at least 0.5 times the largest element) in the first row and permute columns so that it lies in the (1, 1) position. Now add a multiple of the first column to the succeeding columns so that the transformed elements $\tilde{a}_{12} = \tilde{a}_{13} = \cdots = \tilde{a}_{1J_2} = 0$. Continue this procedure until the matrix $\tilde{A}$ has the form of Fig. 3. We perform the same column transformations on B, that is

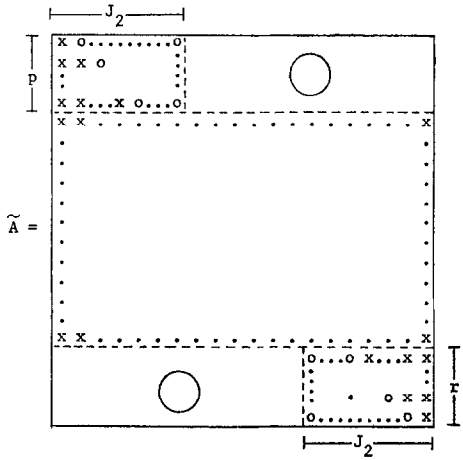

FIG. 3. The A matrix after reduction.

we form $\tilde{B}$ as we are forming $\tilde{A}$. Note that we must have $J_2 > n > p$ in order to have a consistent difference approximation. The form of B does not change, the first p and last r rows still vanish. We test the pivots $\tilde{a}_{ii}$, and if one is too small, then we terminate this procedure and instead try the general reduction. We form matrices $\bar{A}$ and $\bar{B}$ by dropping the first p rows and columns and the last r rows and columns from $\tilde{A}$ and $\tilde{B}$ (see Fig. 4). Since the pivots a_{ii} for

$$1 \leqslant i \leqslant p, \quad J - r + 1 \leqslant i \leqslant J$$

do not vanish, the eigenvalues of $|A + \lambda B| = 0$ are the same as $|\bar{A} + \lambda\bar{B}| = 0$. We know that B is singular. However, there is a good chance that the reduced matrix $\bar{B}$ is nonsingular and, in fact, this is usually the case.

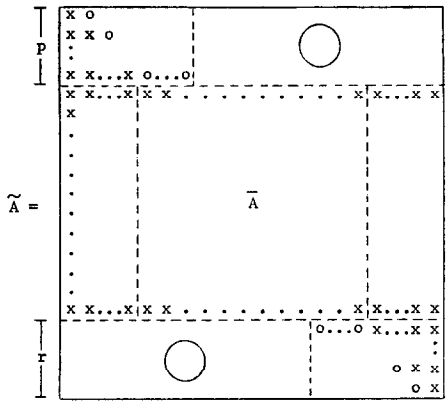

FIG. 4. The $\tilde{A}$ matrix.

We let $\hat{A} = \bar{B}^{-1}\bar{A}$, then the problem is reduced to finding the eigenvalues of $|\hat{A} + \lambda I| = 0$. We test the condition number $\|\bar{B}^{-1}\|\,\|\bar{B}\|$. If this number is too large (say greater than 10^4), we terminate the special reduction and try instead the general reduction. Otherwise, we are in a position to apply the Q-R algorithm to $\hat{A}$ to locate the eigenvalues.

The general reduction proceeds as follows. Using complete pivoting we reduce B to diagonal form. This amounts to finding matrices P and Q such that $B^{(1)} = PBQ$, where $B^{(1)}$ is diagonal. The first step in the reduction is to find the largest element in the matrix B and permute it into the (1, 1) position. Then eliminate the remainder of the first row and column by adding a multiple of the 1st row to the ith row and the 1st column to the jth column, $2 \leqslant i \leqslant J$, $2 \leqslant j \leqslant J$, where J is the order of B. We continue this process until B is reduced to diagonal form.

Now by using row and column permutations, we arrange the diagonal elements in order, $|b_{ii}^{(1)}| \geqslant |b_{i+1,i+1}^{(1)}|$. If $|b_{ii}^{(1)}| > \epsilon$ for $i \leqslant k$, and $|b_{ii}^{(1)}| \leqslant \epsilon$ for $i > k$, then we set $b_{ii}^{(1)} = 0$ for $i > k$. That is, we are assuming the rank of B is k. Denote this matrix by $B^{(1)}$. The proper choice of ϵ is a difficult point which we have simply ignored. This choice should be based on some sort of condition number argument. We have usually set $\epsilon = 10^{-8}$. We will now make a further reduction of A. We have

$$A^{(1)} = PAQ,$$

$$B^{(1)} = PBQ.$$

If the rank of $B^{(1)}$ is equal to its order; that is, if $B^{(1)}$ is nonsingular, then we let

$$\bar{A} = B^{(1)^{-1}}A^{(1)},$$

and the eigenvalue problems $|A + \lambda B| = 0$ and $|\hat{A} + \lambda I| = 0$ are equivalent. In this case, we are ready to use the Q-R algorithm. Otherwise, we must transform A. We denote the rank of $B^{(1)}$ by k, the order of $B^{(1)}$ by J, and let $s = J - k > 0$. We define column transformations on $A^{(1)}$ and $B^{(1)}$ as follows. Permute the columns of $A^{(1)}$ so that the largest element in the $(k + 1)$ row is in the diagonal $(k + 1, k + 1)$ position. Then, by use of elementary column transformations, eliminate all other elements in the $k + 1$ row. Repeat this process so that the matrix $A^{(1)}$ is reduced as shown in Fig. 5. It is clear that the matrix $B^{(2)} = B^{(1)}Q^{(2)}$ has the form shown in Fig. 6 (here $Q^{(2)}$ denotes the product of the elementary column transformations used to reduce $A^{(1)}$).

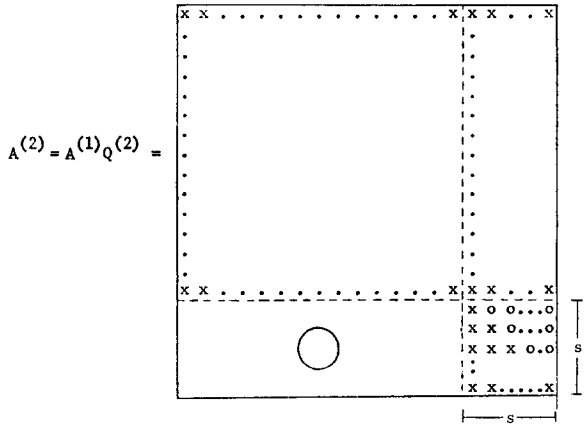

FIG. 5. Reduction of the A matrix.

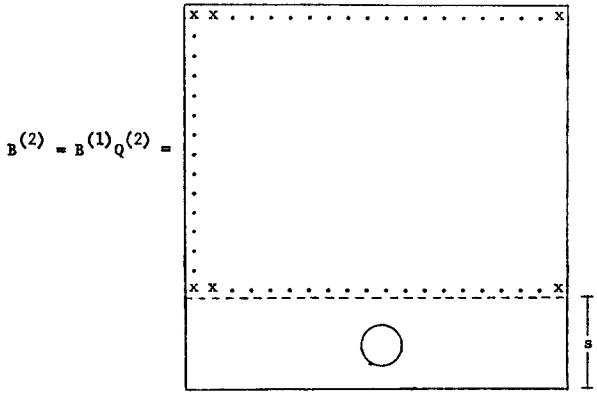

FIG. 6. Reduction of the B matrix.

We let $\bar{A}$ and $\bar{B}$ denote the first k rows and columns of $A^{(2)}$ and $B^{(2)}$. Then the determinants are related by

$$| A^{(2)} + \lambda B^{(2)} | = \left(\prod_{i=k+1}^{J} a_{ii}^{(2)} \right) | \bar{A} + \lambda \bar{B} |.$$

If $| a_{ii} | < \epsilon$ for some $i > k$, then we are unable to complete the reduction and must terminate with an error message. We used the value $\epsilon = 10^{-8}$. This is a rather arbitrary choice, but we experienced no difficulty at this point. We see that the original eigenvalue problem $| A + \lambda B | = 0$ is equivalent to the reduced problem $| \bar{A} + \lambda \bar{B} | = 0$. We now repeat the above reduction process on the smaller matrices $\bar{A}$ and $\bar{B}$. Eventually, we will obtain a matrix $\bar{B}$ whose rank equals its order and the process will terminate.

4. Some Examples

In this section we will consider the result of applying our program to some test cases. The first test case is the equation

$$u^{(4)}(x) + Au^{(2)}(x) + Bu(x) + \lambda[u^{(2)}(x) + Cu(x)] = 0,$$

with boundary conditions $u(0) = u^{(2)}(0) = u(1) = u^{(2)}(1) = 0$. A set of solutions is

$$u(x) = \sin n\pi x,$$

$$\lambda = \frac{[(n\pi)^4 - A(n\pi)^2 + B]}{C - (n\pi)^2}, \qquad n = 1, 2, \ldots .$$

We set $A = 2 + i$, $B = 3 + 2i$, $C = 1 + i$ and used 14 points in the mesh, one point added outside each end of the interval [0, 1] and 12 points within the interval. The error in the eigenvalue for $n = 1$ computed with our matrix method is given in Table I below. We also show the truncation error. The truncation error is obtained by substitution of the exact solution

$$u(x_j) = \sin \pi x_j ,$$

$$\lambda = \frac{\pi^4 - A\pi^2 + B}{C - \pi^2} ,$$

into the finite difference equations.

TABLE I
Error for the Test Case

J	Error in Eigenvalue 14 points	Truncation Error 14 points	Error in Eigenvalue 15 points
4	1.3(−4)	4.(−7)	7.5(−5)
5	3.2(−2)	3.(−8)	3.4(−6)
6	2.3(−7)	8.(−9)	1.3(−7)

The value of J determines the number of points used in the finite difference approximation which is $2J + 1$. We are unable to explain the anomalously large error for $J = 5$. Note that the truncation error does not show this anomalous behavior. We have no explanation for this behavior.

Our next example is the Orr-Sommerfeld equation [1]. This equation is obtained by linearization of the Navier-Stokes equations for viscous fluid flow. We will consider the case of Poiseuille flow between parallel plates. The equation is (where $D = d/dx$)

$$(D^2 - \alpha^2)^2 u - i\alpha R(\bar{U} - \lambda)(D^2 - \alpha^2) u + i\alpha R\bar{U}^{(2)}u = 0,$$

with boundary conditions

$$u(1) = u^{(1)}(1) = 0,$$
$$u^{(1)}(0) = u^{(3)}(0) = 0.$$

The free-stream velocity $\bar{U}(x)$ is given by $\bar{U}(x) = 1 - x^2$. In this problem we wish to find the least stable mode which means we want that eigenvalue whose imaginary part is largest. Since the eigenvector changes more rapidly near $x = 1$ (the plate) than at $x = 0$ (channel center), we use a variable mesh defined by (γ is an input parameter)

$$s(x) = xe^{\gamma x^2 - \gamma}.$$

An equally spaced mesh in s is used to produce a variable mesh spacing in x.

In Table II we consider the effect of a variation in the number of points ($2J + 1$) used in the finite difference approximations (the half-width of the mesh stencil is J). We used $\gamma = 1$ in the above formula for $s(x)$, and 43 points were used in the mesh, 1 point outside each endpoint, and 41 points in the interval. We record all those λ whose imaginary part is positive. The wave number α and the Reynold's number R, which appear in the Orr-Sommerfeld equation, are $\alpha = 1$ and $R = 1 \times 10^4$.

TABLE II
The Eigenvalues as a Function of J

J	Eigenvalues $P = 1\text{-}41\text{-}1, \gamma = 1$
2	$0.23745459 + 0.00280689i$
3	$0.23755142 + 0.00371693i$
4	$0.23752964 + 0.00374248i$
5	$0.23752604 + 0.00373996i$
	$0.9366 + 0.0015i$
6	$0.23752651 + 0.00373932i$
	$0.8499 + 0.0015i$
	$0.9618 + 0.0183i$
7	$0.23752523 + 0.00375362i$
	$0.7483 + 0.0386i$
	$0.8873 + 0.0451i$
	$0.9858 + 0.0261i$

When J is greater than 4, we pick up "growing modes" which are apparently spurious. By "growing modes" we mean eigenvalues with positive imaginary parts. (The function which is a solution of the Orr-Sommerfeld equation determines a solution of the linearized Navier-Stokes equations. This solution contains the factor $e^{i\alpha(x-\lambda t)}$, hence, this solution is a "growing mode" if the imaginary part of λ is positive.) We feel these modes are spurious (by this we mean they do not accurately represent a solution of the differential equation) because they shift wildly with changes in J or changes in the number of mesh points P. If we have a true mode, we should observe convergence as the number of mesh points increases. By comparison of Tables II and III, we see that we get the most accurate results for $J = 6$ (for $P = 1 - 41 - 1$, $\gamma = 1$). However, we do pick up two spurious growing modes. This is one of the drawbacks of this method. We have to run enough cases to be able to reject these spurious modes. We could probably automate this rejection, although we have not done so. It may be best for the user to look carefully at the results, and make his own decision.

In Table III we vary the number of mesh points, P, always using 1 point outside each endpoint. We show all eigenvalues with positive imaginary part.

The eigenvalue obtained by Thomas, using 100 points and a Numerov method (5-point mesh stencil), is $\lambda = 0.2375243 + 0.0037312i$. We obtain somewhat less error using 43 points and $J = 4$.

In Table IV we consider the effect of the parameter γ which appears in the mesh transformation

$$s(x) = xe^{\gamma x^2 - \gamma}.$$

TABLE III
The Dependence on the Mesh Resolution

P	Eigenvalues $J = 4, \gamma = 1$
1–31–1	0.23754393 + 0.00375490i
	0.9806 + 0.00376i
1–41–1	0.23752964 + 0.00374248i
1–51–1	0.23752731 + 0.00374042i
1–61–1	0.23752676 + 0.00373993i
1–71–1	0.23752658 + 0.00373977i
1–81–1	0.23752654 + 0.00373972i
1–98–1	0.23752650 + 0.00373969i

TABLE IV
The Effect of the Variable Mesh Spacing

	$J = 4, P = 1$–41–1	
γ	Eigenvalue	$s'(1)/s'(0) = (1 + 2\gamma)e^{\gamma}$
0.	0.23730744 + 0.00375620i	1.0
1.0	0.23752964 + 0.00374248i	8.2
1.25	0.23752886 + 0.00374148	12.0
	0.9849 + 0.0005i	
1.5	0.23752998 + 0.00374039i	18.0
	0.8243 + 0.0150i	
	0.9678 + 0.0192i	
2.0	0.23760756 + 0.00372823i	37.0
	0.7195 + 0.0944i	
	0.9267 + 0.0736i	

The optimum value for γ for 43 mesh points is apparently between 1.25 and 1.5. Note that we pick up spurious growing modes as γ is increased.

Of course, this rather detailed description of the results applies only to this Poiseuille flow case. No general conclusion about the method can be drawn from these results.

5. Description of a Shooting Method

We will use a shooting method as described by Conte [2]. We consider a linear system of equations

$$dy/dx = Ay, \tag{4}$$

where y is a complex vector of order n and $A = A(x, \lambda)$ is a given function of the real variable x and the complex parameter λ. We let B be a complex $(n - k) \times n$ matrix of boundary conditions at $x = a$, and D a complex $(k \times n)$ matrix of boundary conditions at $x = b$. The problem is to find λ such that the differential equation has a nontrivial solution which satisfies the boundary conditions

$$By(a) = 0 \qquad Dy(b) = 0.$$

The first step in the numerical solution is to establish initial values for $y(x)$ so that we can integrate the equation. We choose an $(n \times n)$ elementary column transformation matrix Q, such that BQ has the form given below.

$$\hat{B} = BQ = \begin{bmatrix} x & 0 & & \cdots & & 0 \\ x & x & 0 & \cdots & & 0 \\ \vdots & & & & & \\ x & x & \cdots & x & 0 & \cdots & 0 \end{bmatrix} \begin{matrix} \top \\ \vdots \\ \bot \end{matrix} n - k$$

$$\underbrace{\qquad\qquad}_{n-k} \qquad\qquad \underbrace{\qquad\qquad\qquad}_{n}$$

We define an $n \times k$ matrix W as below.

$$W = \begin{bmatrix} 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & \\ 1 & 0 & \cdots & \vdots \\ 0 & 1 & & \\ & 0 & \cdots & \vdots \\ \vdots & \vdots & & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix} \begin{matrix} \} \\ \vdots \\ \} \\ \} \\ \vdots \\ \} \\ \vdots \\ \} \end{matrix} \begin{matrix} n - k \\ \\ \\ \\ \\ k \\ \\ \\ \end{matrix}$$

Then if $Y_0 = QW$, we have $BY_0 = 0$. The k columns of Y_0 provide us with k independent initial values to start the integration. If we define $Y(x)$ to be an $(n \times k)$ matrix solution of the differential equation below, then the columns of Y are independent solutions of the original differential equation (4).

$$dY/dx = AY, \qquad Y(a) = Y_0.$$

Note that if $Z = YP$ is a linear combination of the columns of Y [P is an $(k \times k)$ nonsingular matrix], then Z is also a solution, and the boundary condition $BZ = 0$ is satisfied.

The basic idea of this shooting method is to keep the columns of the solution matrix Y approximately orthogonal. We specify points $x_1 < x_2 < \cdots < x_p$ at

which we reorthogonalize the solution. In our program, these points are equally spaced, $p \leqslant 20$, and we require $x_1 = a$ and $x_p = b$. Following Conte, we reorthogonalize at one of these points x_i only if the maximum length of the columns of Y exceeds some input parameter [2]. Thus, starting at x_i, we integrate to x_{i+1}, then test the columns to see if orthogonalization is necessary. If necessary, we orthogonalize the columns of Y by the Gram-Schmidt process. This means we define a matrix P such that the product YP has orthonormal columns. We will describe the process for $k = 2$. We first normalize $y^{(1)}$, where $y^{(1)}$ denotes the first column of Y. That is, define α_1 by

$$\alpha_1 = [y^{(1)} \cdot y^{(1)}]^{-1/2}, \quad \text{where}$$

$u \cdot v$ is the normal complex scalar product $u \cdot v = \sum_{i=1}^{n} u_i \bar{v}_i$. Let

$$\begin{aligned} v^{(1)} &= \alpha_1 y^{(1)}, & \hat{v}^{(2)} &= y^{(2)} - (y^{(2)} \cdot v^{(1)})\, v^{(1)}, \\ \alpha_2 &= (\hat{v}^{(2)} \cdot \hat{v}^{(2)})^{-1/2}, & \beta_2 &= (y^{(2)} \cdot v^{(1)}), \\ v^{(2)} &= \alpha_2 \hat{v}^{(2)}. \end{aligned}$$

Then $v^{(2)} = \alpha_2 y^{(2)} - \alpha_2 \beta_2 \alpha_1 y^{(1)}$ and, thus, P is given by

$$P = \begin{pmatrix} \alpha_1 & -\alpha_1 \alpha_2 \beta_2 \\ 0 & \alpha_2 \end{pmatrix}.$$

Note that even if $y^{(1)} = y^{(1)}(x_{i+1}, \lambda)$ is an analytic function of the complex variable, λ, α_1 is not an analytic function of λ and neither is the matrix YP. It is a C^∞ function of the real and imaginary parts of λ; that is, regard λ as a point in E_2 rather than a complex variable.

We would prefer to have YP an analytic function of λ for reasons that we will shortly discuss. Therefore, we also used the following "quasi-orthogonalization." We define a "quasi-innerproduct"

$$u \circ v = \sum_{i=1}^{n} u_i v_i .$$

Then we let

$$\alpha_1 = \sqrt{y^{(1)} \circ y^{(1)}},$$

where we take the complex square root. If we avoid the branch cuts of this square root, then α_1 is an analytic function of λ. To define the matrix P, we go through the same steps as before using this quasi-innerproduct instead of the standard one.

The result of all this is a solution of the problem

$$dY/dx = AY, \qquad Y(0) = Y_0 P, \qquad a \leqslant x \leqslant b, \tag{5}$$

where P is the product $P = P^{(1)}P^{(2)} \cdots P^{(p)}$ of the orthogonalization matrices (if no orthogonalization is done at x_i, then $P^{(i)} = I$). This requires an integration scheme to solve the equation between x_i and x_{i+1}. We used three schemes, 4-th order Runge-Kutta, Hamming's method, and a more recent 6-th order method due to Rosser [6, 7].

Next, we form the matrix Q defined by

$$Q = DY(b),$$

where D is the matrix of boundary conditions at $x = b$, and $Y(x)$ is the solution of Eq. (5). We note that Q is a function of λ, and we will denote the determinant of Q by $f(\lambda)$,

$$f(\lambda) \equiv \det(Q) \equiv |\, Q \,|\,.$$

In order to obtain a nontrivial solution, we must choose λ so that $f(\lambda) = 0$. We tried two rootfinders to locate these zeros of $f(\lambda)$, Muller's method and Laguerre's method. We did not polish our program enough to obtain a really good shooting method. Our main problem was our inability to obtain a really satisfactory rootfinder. Therefore, our comparison between the matrix method and the shooting method is perhaps suggestive, but certainly not definitive.

6. Results Using the Shooting Method

The essence of this method is the orthogonalization of the solution. Conte suggests integration to the right boundary using an initial guess for the eigenvalue λ. This integration determines points x_i at which orthogonalization is done and matrices $P^{(i)}$, which orthogonalize the solution vector. These points x_i and matrices $P^{(i)}$ are then fixed and used for later integrations at different values of λ. This procedure insures that the determinant $f(\lambda)$ is an analytic function of λ. We were unable to make this procedure work for the Poiseuille flow case. It turns out that the matrices $P^{(i)}$ must be quite accurate, in one case to 5 digits, in order that the resulting solution vectors be orthogonal to one digit. If the vectors do not remain orthogonal, then catastrophic growth occurs. If the x_i and $P^{(i)}$ are chosen at $\lambda = 0.2000 + 0.003i$ and then these values of x_i and $P^{(i)}$ are used for $\lambda = 0.1999 + 0.003i$, we found the solution vectors grew to magnitude $\sim 10^{19}$ at the right boundary. For $\lambda = .2 + .003i$ the solution vectors had reasonable magnitude. Therefore, we recomputed the quantities x_i, $P^{(i)}$ for each value of λ even though this results in a nonanalytic determinant $f(\lambda)$. We seemed to obtain better results with the quasi-innerproduct described in Section 5. A comparison between the standard and quasi-innerproduct is given in Table V. In these cases,

TABLE V
The Effect of the Innerproduct on the Laguerre Rootfinder
(Poiseuille Flow, $\alpha = 1$, $R = 10{,}000$)

Initial Guess	Standard Innerproduct			Quasi-Innerproduct		
	Root	Iterations	$\lvert f/f' \rvert$	Root	Iterations	$\lvert f/f' \rvert$
$\lambda_0 = 0.0 + 0.0i$	$0.35 - 0.12i$	8	5.(−13)	$0.23751 + 0.00376i$	5	2.(−11)
	$0.19 - 0.18i$	8	1.(−13)			
	$-0.63 - 0.95i$	19	2.(−10)			
	$-0.29 - 1.1i$	23	5.(−10)			
	$-0.72 - 0.85i$	30[a]	1.(−10)			
$\lambda_0 = 0.5 + 0.05i$	$0.91 - 0.09i$	11	9.(−14)	$0.23751 + 0.00376i$	4	5.(−12)
	$0.85 - 0.15i$	12	7.(−13)			
	$0.85 - 0.17i$	14	3.(−12)			
	$0.79 - 0.20i$	13	6.(−9)			
	$0.71 - 0.21i$	15	1.(−10)			
	$0.63 - 0.21i$	20	6.(−9)			
	$0.48 - 0.21i$	20	1.(−9)			
	$-0.16 - 0.98i$	30[a]	9.(−3)			

[a] Rootfinder failed to satisfy the convergence test.

we used a predictor corrector (Hamming's method) with a step size $h = 0.0078$. We report results for two values of the initial guess for the eigenvalue λ : $\lambda_0 = 0$ and $\lambda_0 = 0.5 + 0.05i$. For $\lambda_0 = 0$, the routine using the quasi-innerproduct found the least stable mode on the first attempt; the standard product routine failed to satisfy the convergence test on the fifth root and never did find the least stable eigenvalue.

We indicate the ratio $|f(\lambda)/f'(\lambda)|$ evaluated at the root in order to give some idea of the accuracy of the root. The derivative is approximated by a symmetric finite difference quotient. We also give the number of iterations required by the Laguerre rootfinder (there must be three function evaluations for each iteration).

In Table VI we compare the Muller and Laguerre rootfinder for various initial guesses λ_0. The Laguerre seems to be superior. In all these cases we used Hamming's method, with the quasi-innerproduct.

A comparison of computing time between the shooting method and the matrix method is of course impracticable because of the effect of the initial guess on speed of convergence. However, even granted the availability of a reasonably good initial guess, we were not able to obtain the expected time advantage of the shooting method because of the small step size we were forced to use. Hamming's method ($h = 0.0078$) used in the shooting scheme required 5.3 sec of computer time to

TABLE VI
A Comparison of the Rootfinders

Initial Guess	Laguerre	Muller
$\lambda_0 = 0.$	Found $\lambda = 0.23751 + 0.00376$ in 5 iterations	Found $\lambda = 0.23751 + 0.00376$ in 15 iterations
$\lambda_0 = 0.5 + 0.05i$	Found $\lambda = 0.23751 + 0.00376$ in 4 iterations	Failed to find any roots
$\lambda_0 = -1. - 0.1i$	Found 6 roots; none were the least stable mode; failed on 7-th root	Failed to find any roots
$\lambda_0 = 1. + 0.1i$	Found 3 roots; failed on 4-th root; 3rd root was least stable mode $\lambda = 0.23751 + 0.00376i$	Failed to find any roots

yield the value $\lambda = 0.2375097 + 0.0037606i$. The Q-R matrix method required 10.6 sec (using 41 mesh points) to yield the better approximation

$$\lambda = 0.2375296 + 0.0037424i.$$

References

1. R. Betchov and W. Criminale, "Stability of Parallel Flows," Academic Press, New York, 1967.
2. S. Conte, The numerical solution of linear boundary value problems, *SIAM Rev.* **8** (1966), 301–321.
3. J. Gary, Hyman's method applied to the general eigenvalue problem, *Math. Comp.* **19** (1965), 314–316.
4. R. Hamming, "Numerical Methods for Scientists and Engineers," McGraw-Hill, New York, 1962.
5. M. Osborne, Numerical methods for hydrodynamic stability problems, *SIAM J. App. Math.* **15** (1967), 539–558.
6. A. Ralston and H. Wilf (Eds.), "Numerical Integration Methods for the Solution of Ordinary Differential Equations," Mathematical Methods for Digital Computers, Vol. 1, pp. 95–109, Wiley, New York, 1960.
7. J. Rosser, A Runge-Kutta for all seasons, *SIAM Rev.* **9** (1967), 417–452.
8. J. Stengl and C. Isaacs, "A New Solution Method for the Determinantal Equation of a Matrix Polynomial," Proc. of 21st ACM Conf., (1966), pp. 21–27.
9. I. Tarnove, Determination of eigenvalues of matrices having polynomial elements, *J. SIAM* **6** (1958), 163–171.
10. L. Thomas, "The Stability of Plane Poiseuille Flow," *Phys. Rev.* **91** (1953), 780–783.
11. J. Wilkinson, "The Algebraic Eigenvalue Problem," Clarendon Press, Oxford, 1965.
12. J. Gary and R. Helgason, "ODEIG — Program to solve eigenvalue problems in the form of two-point boundary value problems," NCAR Computing Facility documentation, Boulder, Colorado (November 1968).