Journal of Computational and Applied Mathematics 34 (1991) 201–209
North-Holland

Eigensolutions of boundary value problems using inverse iteration

David R. Fearn
Department of Mathematics, University of Glasgow, University Gardens, Glasgow, United Kingdom G12 8QW

Received 6 April 1990

Abstract

Fearn, D.R., Eigensolutions of boundary value problems using inverse iteration, Journal of Computational and Applied Mathematics 34 (1991) 201–209.

Matrix eigenvalue problems arise when the differential operators in a system of ordinary or partial differential equations are replaced by finite-difference operators. We describe the use of the method of inverse iteration to solve such eigenvalue problems. The key to the success of this method is that it can take full advantage of the band structure of the matrix, resulting in a very considerable savings in storage and CPU-time compared with other matrix methods. For ordinary differential equations, the time taken is proportional to the number of grid points chosen. To illustrate the method, we solve the Orr–Sommerfeld problem, using both second- and fourth-order difference schemes. For a given accuracy of solution, the latter requires a similar CPU-time to shooting with orthonormalization. We show that the inverse iteration method has no trouble coping with very stiff problems.

Keywords: Boundary value problems, eigenvalues, inverse iteration, stability problems, Orr–Sommerfeld equation, stiff problems.

1. Introduction

Many problems in mathematical physics require the solution of an eigenvalue problem. For example, in linear stability calculations, the eigenvalue is the growth rate p of an instability. For some problems, p will be real (exchange of stabilities) but more generally p is complex; the imaginary part being the frequency of the instability. Often, we are interested in the solution with the largest $\mathrm{Re}(p)$; the most unstable mode. When p is complex, the use of any method, such as shooting, that finds only a single eigensolution leaves open the question: has the most unstable mode been found? To ensure that the mode with the largest $\mathrm{Re}(p)$ is identified, it is usually necessary to find all the eigenvalues of the system and pick out the required one. To do this, the governing ordinary or partial differential equations must be discretized (for example by using finite differences), giving a matrix eigenvalue problem. This can then be solved for all eigenvalues by methods based on, for example, the QR or LR algorithms (see [9]). The drawback of this approach is the limitation it places on numerical resolution. Methods that find all the eigenvalues require the full matrix to be stored and worked with. Storage is proportional to N^2

and CPU-time of order N^3 where N is the number of grid points (or equivalent) used in the discretization. For ordinary differential equations this may not be too restrictive, but for partial differential equations, the storage requirements place a severe limitation on numerical resolution.

One approach to overcoming the problem is to start with a modest value of N and use a full-matrix method. This finds all the eigenvalues, so the required mode (for example, that with the largest $\mathrm{Re}(p)$) can be identified. Since N is not large, the eigensolution found will not be an accurate solution of the original differential equation. To improve the accuracy, we must use a more efficient method that permits large values of N to be used. Such a method will typically only give a single eigenvalue (and the corresponding eigenvector) and requires as an input parameter, an estimate of the required eigenvalue. The eigenvalue obtained from the full-matrix method at modest N is of course the estimate to use in the more efficient method at larger values of N. One example of a more efficient method is shooting, but this would require a recoding of the problem. An alternative is to use a method that uses the same matrix as before, but in a more efficient manner. Such a method is inverse iteration, (see, for example, [10]). When discretized using finite differences, an ordinary or partial differential equation produces a banded matrix; all the entries in the matrix outside a band of width L, centred approximately on the leading diagonal, are zero. The method of inverse iteration can be coded to work only with the nonzero band, resulting in considerable savings in storage and CPU-time. The storage requirements are $O(NL)$ and the CPU-time $O(NL^2)$. For an ordinary differential equation, L is fixed, independent of N, so both storage and CPU-time are linear in the number of grid points.

In this paper, we use the Orr–Sommerfeld problem (see, for example, [5]) to test the method of inverse iteration. Both second- and fourth-order difference schemes are used, and the results compared with those obtained by a shooting method with orthonormalization [2,4]. The Orr–Sommerfeld problem is a fourth-order ordinary differential equation. With second-order finite differences, the band width $L = 5$, and with fourth-order differences, $L = 11$. In Section 3 we show that, for a given accuracy of solution, the fourth-order difference scheme requires a much smaller number N of grid points than the second-order scheme. This more than compensates for the increased band width; the CPU-time required for a given accuracy being less using fourth-order differences.

A direct comparison is made between inverse iteration using a fourth-order difference scheme and shooting using a fourth-order Runge–Kutta scheme. The two methods are very close in the time taken to find a solution of given accuracy. From the timing point of view, there is therefore little to choose between the two. There are two good reasons for choosing inverse iteration. Firstly, in cases where a full-matrix method has had to be used to find the most unstable mode, the coding required to generate the matrix from the differential equations has already been written. Only minor modifications are required to store only the nonzero band of the matrix. Secondly, and more importantly, inverse iteration can cope with stiff problems, without modification. In cases where standard shooting fails and techniques such as orthonormalization [2–4] have to be incorporated, inverse iteration works well, provided only that sufficient grid points have been incorporated to resolve such localized features as boundary layers or critical layers. In Section 4 we solve the Orr–Sommerfeld problem for the stability of plane Poiseuille flow at high Reynolds number R and compare the results found using inverse iteration with those obtained by alternative methods. It is well known (see [5]) that solutions of the Orr–Sommerfeld problem contain a critical layer of width $O(R^{-1/3})$. We show that inverse iteration has no trouble coping with Reynolds numbers as high as $R = 10^9$.

In Section 5 we discuss the application to partial differential equations and vectorization of the program.

2. Application of the method

Given a linear eigenvalue problem of the form

$$L\phi = p\phi, \tag{1}$$

where L is a linear differential operator, replacing differential operators by difference operators transforms (1) into a matrix eigenvalue problem of the form

$$A\phi = p\phi. \tag{2}$$

If A is an $N \times N$ matrix, then (2) has eigenvalues p_j and eigenvectors ϕ_j satisfying

$$A\phi_j = p_j\phi_j, \quad j = 1, \ldots, N. \tag{3}$$

Given some guess ψ for the eigenvector and q for the eigenvalue, the inverse iteration scheme (see, for example, [10])

$$\psi^{(m)} = (A - qI)^{-1}\psi^{(m-1)}, \quad m = 1, 2, \ldots, \qquad \psi^{(0)} = \psi, \tag{4}$$

converges to the eigenvector ϕ_i whose eigenvalue p_i is closest to q, i.e., the eigenvalue satisfying

$$|p_i - q| < |p_j - q|, \quad j = 1, \ldots, i-1, i+1, \ldots, N. \tag{5}$$

The rate of convergence depends on the relative magnitudes of $|p_i - q|$ and $|p_j - q|$. If $|p_i - q| \ll |p_j - q|$ for all $j \neq i$, then scheme (4) converges rapidly to ϕ_i. A good quess q for the required eigenvalue is therefore necessary for the efficient working of this method. On the other hand, the scheme is fairly insensitive to the guess ψ for the eigenvector. If no good estimate for the eigenvector is available, then choosing $\psi = (1, 1, 1, \ldots, 1)^{\mathrm{T}}$ suffices.

The technique can be easily extended to the generalized eigenvalue problem

$$A\phi = pB\phi, \tag{6}$$

when the iterative scheme (4) is replaced by

$$\psi^{(m)} = (A - qB)^{-1}B\psi^{(m-1)}, \quad m = 1, 2, \ldots, \qquad \psi^{(0)} = \psi. \tag{7}$$

The scheme (4) or (7) is terminated when two successive iterates $\psi^{(m)}$ and $\psi^{(m-1)}$ are parallel to within some tolerance. The magnitude of $\psi^{(m)}$ exceeds that of $\psi^{(m-1)}$ by a factor $(p_i - q)^{-1}$ since $(A - qB)^{-1}B\phi_i = (p_i - q)^{-1}\phi_i$. Hence, calculating the relative magnitude of $\psi^{(m)}$ compared with $\psi^{(m-1)}$ gives the correction $(p_i - q)$ to the guess q for the eigenvalue. Thus, the method of inverse iteration gives successive iterative approximations to the required eigenvector. Only when the required accuracy has been reached, is the eigenvalue $p_i = q + (p_i - q)$ calculated.

As discussed in Section 1, the matrices A and B are banded with all entries outside bands of widths L_A and L_B respectively equal to zero. Only the nonzero bands need to be stored and the scheme (7) can be rewritten as

$$(A - qB)\psi^{(m)} = B\psi^{(m-1)}. \tag{8}$$

This can be solved using an LU decomposition of $(A - qB)$. The LU decomposition only needs to be performed once (since the estimate q for the eigenvalue is not updated at each iteration) and no extra storage is required since L and U can be stored in the array originally occupied by $A - qB$.

3. Second- and fourth-order difference schemes

To illustrate the use of the method of inverse iteration we solve the Orr–Sommerfeld problem for the linear stability of the parallel shear flow $U(z)$. The Orr–Sommerfeld equation (see [5]) may be written in the form

$$\left[-(\mathrm{i}\alpha R)^{-1}(\mathrm{D}^2 - \alpha^2)^2 + U(\mathrm{D}^2 - \alpha^2) - U''\right]\phi = c(\mathrm{D}^2 - \alpha^2)\phi, \tag{9}$$

where D denotes $\mathrm{d}/\mathrm{d}z$, α is the wavenumber in the direction of the flow, αc is the frequency of the instability, and R is the Reynolds number which is a nondimensional measure of the flow speed. To permit comparisons with other work, we choose the plane Poiseuille flow $U = 1 - z^2$, $-1 \leqslant z \leqslant 1$, and look for solutions symmetric about the mid-layer $z = 0$. The equation (9) is therefore solved in the half-layer $0 \leqslant z \leqslant 1$, subject to the boundary conditions

$$\mathrm{D}\phi = \mathrm{D}^3\phi = 0 \quad \text{at } z = 0, \qquad \phi = \mathrm{D}\phi = 0 \quad \text{at } z = 1. \tag{10}$$

The interval [0, 1] is divided into N equal subintervals and $\phi(z)$ approximated by its values at the nodes:

$$z_i = ih, \quad h = 1/N, \qquad \phi_i = \phi(z_i). \tag{11}$$

The equation (9) is evaluated at the $N - 1$ interior nodes z_i, $i = 1, \ldots, N - 1$. The highest derivative in (9) is D^4. The finite-difference approximations for derivatives can be found, for example, in [1, Table 25.2]. Using second-order central finite differences, we obtain $N - 1$ equations for the $N + 3$ unknowns ϕ_i, $i = -1, \ldots, N + 1$. The four unknowns ϕ_{-1}, ϕ_0, ϕ_N, ϕ_{N+1} can be eliminated using the boundary conditions (10). Central differences can be used except for $\mathrm{D}^3\phi = 0$, where a forward difference is required to avoid including a further unknown ϕ_{-2}. Equation (9) has now been reduced to a matrix eigenvalue problem of the form (6), where A has band width 5 and B band width 3.

For $R = \mathrm{O}(1)$, (9) is well behaved, and an accurate solution can be obtained using a modest number N of subintervals. When R is large, the highest derivatives only become important at the boundaries and in a critical layer of width $\mathrm{O}(\alpha R)^{-1/3}$ centred on $U = c$ (see [5]). Thus the Orr–Sommerfeld problem at high Reynolds numbers represents a rigorous test of any numerical method. Provided that N is chosen sufficiently large to resolve the critical layer, inverse iteration works well. No special modifications are required at high Reynolds number.

In Table 1 are listed some results for the case $R = 10000$, $\alpha = 1$. The solution, correct to five decimal places is $c = 0.23753 + 0.00374\ \mathrm{i}$ [2]. The results in Table 1 demonstrate that inverse iteration easily copes with high Reynolds numbers, but the number of subintervals required to obtain an accurate solution is large. One reason for this is of course the need to resolve the critical layer, but in this example the critical layer has width $\mathrm{O}(\alpha R)^{-1/3} = \mathrm{O}(0.05)$, so should be adequately resolved in all the results in Table 1. The slow convergence to the result as N is increased is because we have used a second-order difference scheme (relative error $\mathrm{O}(N^{-2})$). This

Table 1
Some results for the Orr–Sommerfeld problem for plane Poiseuille flow with $R = 10000$ and $\alpha = 1$; the initial guess for the eigenvalue was $c = 0.23753$ and NIT iterations were required to reach the required accuracy; the scheme was terminated when the largest entry of $\hat{\phi}^{(m)} - \hat{\phi}^{(m-1)}$ has a magnitude of less than 10^{-8}; the vector $\hat{\phi}^{(m)}$ is $\phi^{(m)}$ normalized such that its entry with the largest magnitude is unity; the CPU-time taken was measured on an IBM 370/168

N	c	NIT	CPU-time (seconds)
500	0.237505 + 0.003736 i	6	1.3
1000	0.237521 + 0.003739 i	6	2.3
2000	0.237525 + 0.003739 i	6	4.2
4000	0.237526 + 0.003739 i	6	7.8

becomes inefficient when high accuracy is required because of the large values of N that need to be used. Techniques such as shooting typically use fourth-order schemes (relative error $O(N^{-4})$), so for a fair comparison with shooting methods we repeat our calculations using fourth-order finite differences.

The use of fourth-order differences is not quite so straightforward as second-order differences. Were we to evaluate (9) at the $N-1$ interior nodes using central differences, we would obtain $N-1$ equations in the $N+5$ unknowns ϕ_i, $i = -2, \ldots, N+2$. The boundary conditions can determine only four of these, leaving two unknowns undetermined. The solution to this problem is to use forward differences when evaluating (9) at $z = z_1$ and backward differences at $z = z_{N-1}$. Forward and backward differences also need to be used for the boundary conditions (10). Results corresponding to those of Table 1 are shown in Table 2. The time required for a given value of N is greater for the fourth-order difference scheme because the band widths of the matrices are larger, A having band width 11. This increased time is more than compensated for by the greater accuracy of the fourth-order scheme. For a given accuracy of solution, the fourth-order scheme requires a much smaller value of N and has correspondingly smaller storage and CPU-time requirements.

It is clear that the extra effort involved in using a fourth-order difference scheme is worthwile. We proceed to compare the results of the inverse iteration method with alternative methods of solving the Orr–Sommerfeld problem.

Table 2
As Table 1 but using a fourth-order difference scheme

N	c	NIT	CPU-time (seconds)
200	0.237532 + 0.003746 i	6	1.1
400	0.237527 + 0.003740 i	6	1.8
600	0.237527 + 0.003740 i	6	2.5
800	0.237527 + 0.003740 i	6	3.2
1000	0.237526 + 0.003740 i	6	3.9

4. Comparison with alternative methods

Detailed comparisons were carried out with the method of shooting with orthonormalization [2]. Orthonormalization is required because simple shooting fails for stiff problems, such as the Orr–Sommerfeld problem at high Reynolds numbers. Both methods use fourth-order difference schemes and take a time proportional to N, the number of subintervals, so they are directly comparable. For $R = 10^6$, $\alpha = 1$ the solution of (9) is $c = 0.06659252 - 0.01398327$ i [2]. For both methods, we took the intial guess at the eigenvalue to be $0.0666 - 0.0140$ i (see later) and the results are given in Tables 3 and 4. The calculations were performed on the same machine (an IBM 370/168), under (as near as possible) identical conditions. It can be seen that for comparable accuracy, and for the particular case solved, inverse iteration is marginally faster but requires a value of N approximately 1.5 times that required by shooting with orthonormalization. Since the calculations of Table 4 were performed, some rewriting of the inverse iteration code has led to savings of about 20% in time taken. Inverse iteration has greater storage requirements than orthonormalization but for ordinary differential equations this is not a serious restriction on modern computers. The two methods are therefore comparable in performance. For a problem suitable for solution using inverse iteration, the advantages discussed in Section 1 are good reasons for choosing inverse iteration.

To give some idea of how far inverse iteration can be pushed, we compared results for $R = 10^9$, $\alpha = 1$ with those of Davey [4] who uses both standard orthonormalization and an automatic orthonormalization method. Using $N = 24000$, we found $c = 0.00656631 - 0.00166002$ i. Davey quotes $0.00656630 - 0.00166002$ i, correct to four significant figures. The number of subintervals used is comparable. Davey tabulates values of the eigenfunction in his Table III. Our results match his to five significant figures except for the last few points close to $z = 1$ where the agreement is only to four decimal places, see Table 5. Davey [4] states that his solutions for the eigenvalue are correct to four significant figures. Presuming that the eigenfunction is to the same accuracy, our results agree to the accuracy calculated. Davey [4] claims that his method has been used "on a very stiff Orr–Sommerfeld problem to calculate eigenfunctions which no other method has been able to produce". Inverse iteration has reproduced these results.

This comparison emphasizes the power of the inverse iteration method. It can cope with very stiff problems without modification, only requiring a sufficient number of subintervals to resolve all features of the eigenfunction. It is interesting to note that the LU decomposition has been performed on matrices of up to 24000 rows without their being any evidence of problems caused by rounding errors.

Table 3
Results for the Orr–Sommerfeld problem with $R = 10^6$, $\alpha = 1$; the method used was shooting with orthonormalization and the calculations were carried out by A. Davey; the initial guess for the eigenvalue was $c = 0.0666 - 0.0140$ i and the error tabulated is the magnitude of the difference from $c = 0.06659252 - 0.01398327$ i; the CPU-time was measured on an IBM 370/168

N	c	Error	CPU-time
800	$0.06659223 - 0.01398184$ i	$146 \cdot 10^{-8}$	3.0
900	$0.06659211 - 0.01398246$ i	$91 \cdot 10^{-8}$	3.4
1000	$0.06659221 - 0.01398269$ i	$66 \cdot 10^{-8}$	3.8

Table 4
As Table 3 but using inverse iteration with fourth-order differences; for a given value of N, inverse iteration has a larger error than shooting; the values of N ($N = 1350, 1500$) have been chosen to give errors comparable with those for $N = 900, 1000$ in Table 3

N	c	Error	CPU-time	NIT
1000	0.06659520 − 0.01398211 i	$292 \cdot 10^{-8}$	2.5	3
1350	0.06659336 − 0.01398293 i	$91 \cdot 10^{-8}$	3.2	3
1500	0.06659308 − 0.01398305 i	$60 \cdot 10^{-8}$	3.6	3

One criticism that could be made of the calculations presented above for the inverse iteration method is that the estimates used for the eigenvalues are very close to the required eigenvalues. This is the case because we have chosen our estimates to be the same as those used by Davey. This leaves open the question of how well does inverse iteration work when the estimate is much poorer. How far the estimates can be from the eigenvalue and still converge can be judged from the discussion of Section 2. The method always converges to the eigenvalue closest to the estimate, so how accurate an estimate has to be depends on the distribution of eigenvalues. For the solution in Table 4 we repeated our calculation for different estimates, see Table 6. As can be seen the method can still work well even when the estimate is poor. When the number of inverse iterations becomes large, the strategy of performing the LU decomposition once and calculating the eigenvalue at the end of the many iterations becomes inefficient. Performing some number of iterations, updating the estimate for the eigenvalue (which requires performing the LU decom-

Table 5
Selected values of the eigenfunction for $\alpha = 1$, $R = 10^9$, for comparison with [4, Table III]

z	ϕ_r	ϕ_i
0	1	0
0.90	0.479061	0.001681
0.91	0.463503	0.001738
0.92	0.447431	0.001797
0.93	0.430794	0.001862
0.94	0.413527	0.001932
0.95	0.395547	0.002011
0.96	0.376731	0.002102
0.97	0.356894	0.002212
0.98	0.335707	0.002359
0.99	0.312404	0.002616
0.991	0.309883	0.002658
0.992	0.307324	0.002709
0.993	0.304603	0.002863
0.994	0.301605	0.002170
0.995	0.302994	0.001768
0.996	0.302934	0.018106
0.997	0.261233	0.043260
0.998	0.171716	0.029008
0.999	0.066728	−0.005589
1	0	0

Table 6
As Table 4 but using $N = 1500$ for all calculations; the solution for c is as in Table 4; here we tabulate the initial guess for c and the number of iterations required to converge to this solution; the calculations were performed on an IBM 3090/150E.

Estimate for c	CPU-time	NIT
$0.06 - 0.01\,i$	1.14	11
0.03	4.28	63
0.09	3.60	52
$0.0 + 0.1\,i$	5.90	89

position again), then proceeding with inverse iterations would result in a smaller number of iterations being required. The trade-off point between the extra LU decomposition and the reduced number of iterations will depend on the system being solved.

5. Conclusions

We have shown how using the method of inverse iteration to solve the matrix eigenvalue problems that result from discretizing boundary value problems compares favourably with alternative methods of solving such problems. Inverse iteration works well even with very stiff problems. Other examples of the application of inverse iteration to problems with critical layers and boundary layers can be found in [6,7].

The principle reason for the performance of the method described here is its ability to make full use of the banded structure of the matrix. For ordinary differential equations (or systems of ODEs) the band width is fixed, independent of the number N of subintervals. The storage and time requirements are then proportional to N. For low-order systems, the band width is small and no benefit can be derived from vectorizing the LU decompostion code. This is because the recursive nature of the algorithm means that vectorization can only be performed on the innermost loop which runs over the band width. For the Orr–Sommerfeld problem, this is only 11, and calculations on an IBM 3090/150E-VF were some 20% slower in vector than in scalar because of the small loop counts. For higher-order ordinary differential equations (that give matrices with larger band width) there may be a small benefit from using vectorized code. The main benefit from vectorization comes when the method is applied to partial differential equations. For these the band width is dependent on the numerical truncation. The main limitation on the resolution is typically the storage available rather than the CPU-time required. This is particularly true when the benefits of vectorization are used. On a problem of magnetic field stability (see [8]), vectorized code ran $2\frac{1}{2}$ to 3 times faster than scalar with 60–70% of CPU-time being in the vector facility on an IBM 3090/150E-VF.

Acknowledgements

I would particularly like to thank Dr A. Davey for many discussions about this work and for his performing the calculations of Table 3. The inverse iteration program is based on one written

by Dr G.O. Roberts. The original calculations were performed on the IBM 370/168 of the University of Newcastle-upon-Tyne. The most recent calculations have used the IBM 3090/150E-VF supplied to the University of Glasgow by IBM as part of the Kelvin Project. Thanks are due to John Hague of IBM for vectorizing the inverse iteration subroutines.

References

[1] M. Abramowitz and I.A. Stegun, *Handbook of Mathematical Functions* (Dover, New York, 1965).
[2] A. Davey, A simple numerical method for solving Orr–Sommerfeld problems, *Quart. J. Mech. Appl. Math.* **26** (1973) 401–411.
[3] A. Davey, On the removal of the singularities from the Ricatti method, *J. Comput. Phys.* **30** (1979) 137–144.
[4] A. Davey, An automatic orthonormalisation method for solving stiff boundary-value problems, *J. Comput. Phys.* **51** (1983) 343–356.
[5] P.G. Drazin and W.H. Reid, *Hydrodynamic Stability* (Cambridge Univ. Press, Cambridge, 1981).
[6] D.R. Fearn, Hydromagnetic waves in a differentially rotating annulus I. A test of local stability analysis, *Geophys. Astrophys. Fluid Dynamics* **25** (1983) 65–75.
[7] D.R. Fearn, Hydromagnetic waves in a differentially rotating annulus II. Resistive instabilities, *Geophys. Astrophys. Fluid Dynamics* **30** (1984) 227–239.
[8] D.R. Fearn and M.R.E. Proctor, Hydromagnetic waves in a differentially rotating sphere, *J. Fluid Mech.* **128** (1983) 1–20.
[9] G. Peters and J.H. Wilkinson, Eigenvectors of real and complex matrices by LR and QR-triangularisations, in: J.H. Wilkinson and C. Reinsch, Eds., *Handbook for Automatic Computation, Vol. 2, Linear Algebra* (Springer, Berlin, 1971) 370–395.
[10] G. Peters and J.H. Wilkinson, The calculation of specified eigenvectors by inverse iteration, in: J.H. Wilkinson and C. Reinsch, Eds., *Handbook for Automatic Computation, Vol. 2, Linear Algebra* (Springer, Berlin, 1971) 418–439.