Iterative Algorithm for Computing the Eigenvalues

LILJANA FERBAR
Faculty of Economics
University of Ljubljana
Kardeljeva pl. 17, 1000 Ljubljana
SLOVENIA

Abstract: - We consider the eigenvalue problem $Hx = \lambda x$, where H is a real symmetric matrix. When eigenvalues of symmetric matrices are computed it is generally expected that they will be computed with an error bound proportional to the product of machine precision and the norm of the matrix. In particular, tiny eigenvalues are usually not computed to high relative accuracy. We propose an iterative refinement of the eigensolution computed by a fast method. Our algorithm usually gives the eigensolution with high relative accuracy and it is more efficient than accurate Jacobi type methods.

Key-Words: - Symmetric eigenvalue problem, Fast method, Jacobi method, Stopping criterion, Accuracy, Efficiency

1 Introduction

In this paper we consider the eigenvalue problem $Hx = \lambda x$. The scalar λ is the eigenvalue and the vector x is the corresponding eigenvector of the matrix H. If H is symmetric or Hermitian matrix of order n, then H has exactly n real eigenvalues, and the corresponding eigenvectors span the basis of the n-dimensional space. More precisely, for a symmetric matrix H we have

$$Q^T HQ = \Lambda, \qquad (1)$$

where $\Lambda = \mathrm{diag}(\lambda_i)$ is a diagonal matrix with eigenvalues of H on the diagonal, and Q is orthogonal matrix whose columns are the corresponding eigenvectors.

Solution of many problems in technical applications is reduced to solving eigenvalue problems. Thus, these problems attract considerable attention and represent one of the most important areas of numerical linear algebra.

The first method for solving eigenvalue problem for symmetric matrices is the Jacobi method [10, 6], which dates back in 1846. The Jacobi method constructs a sequence of matrices

$$H_1 = H, \quad H_{k+1} = R_k^T H_k R_k,$$

which converges to the eigenvalue matrix Λ, while the sequence of products $R_1 R_2 \cdots R_k$ converges to the eigenvector matrix Q. Matrices R_k are the orthogonal plane rotation matrices chosen to annihilate one off-diagonal element of the matrix H_k. Due to the finite arithmetic of the computer this infinite iterative procedure stops after a finite number of steps.

In 1960-ties the QR methods [10, 6] are being developed by many authors. These methods first reduce the symmetric matrix H to tridiagonal matrix T by using orthogonal similarity transformations, and then use QR iterations to solve the eigenvalue problems for the matrix T. Although both methods require $O(n^3)$ floating-point operations, the QR methods are about five times faster than Jacobi type methods.

The divide-and-conquer (D&C) method, which is particularly efficient for tridiagonal matrices, is very suitable for multi-processor computers. The method first partitions the starting matrix into blocks, then solves the smaller eigenvalue problems, and finally connects all the solutions. We conclude that the choice of the method depends upon the structure and the size of the matrix, on the requirements for speed and accuracy, and the available hardware.

Using the computers in solving eigenvalue problems has lead to two aspects of research: speed and accuracy. Due to the need of solving larger and larger problems, the first aspect of research is

finding faster algorithms and the analysis of their speed of convergence. The second is the question of accuracy: how many accurate digits does the computed eigenvalue have?

In applications four kinds of errors appear: errors of the model, errors in data, errors in storing the matrix into the computer, and the errors generated by the computational method. When storing the matrix H into the computer, instead of the element H_{ij} we store the element

$$H_{ij} + \delta H_{ij}, \; |\delta H_{ij}| \leq \varepsilon \, |H_{ij}|, \tag{2}$$

where ε is the machine precision, $\varepsilon \approx 10^{-8}$ (single precision) or $\varepsilon \approx 10^{-16}$ (double precision). Therefore, the last stored digit need not be correct and instead of H we store some $H + \delta H$. The condition is defined as the number κ, which tells us how many times does the error in original data increase. If λ_i is the i-th largest eigenvalue of the matrix H, and $\lambda_i + \delta\lambda_i$ is the i-th largest computed eigenvalue, then the answer to the question about accuracy generally has the form

$$|\delta\lambda_i| \leq \kappa \cdot \|\delta H\| \cdot |\lambda_i|. \tag{3}$$

The condition κ depends on the matrix, but also on the computational method, which we use. We know that accuracy of the computed eigenvalue depends on the following: is the matrix "well-behaved", this is, do small relative changes in matrix elements cause small relative changes in eigenvalues, and if this is the case, which algorithm computes eigenvalues with this accuracy. In general, to answer the first question an appropriate perturbation theory for the given type of problem needs to be developed, while the answer to the second question is given by the numerical analysis of the algorithm.

Many authors have noticed that for some problems different methods give answers with widely varying accuracy. For example, in 1968 Rosanoff et al. [7] noticed that the Jacobi method often computed tiny eigenvalues much more accurate than the QR method. The authors had many excellent observations and gave interesting explanations for facts, which were much later established with complete mathematical rigor. In 1980-ties many articles appear and the intensive research is still going on.

The symmetric (Hermitian) eigenvalue problem was analyzed by Barlow and Demmel [1] for scaled diagonally dominant matrices, by Demmel and Veselić [2] for positive definite matrices and by Veselić and Slapničar [9] for indefinite matrices. They proved that different types of matrices written above are "well-behaved" matrices.

2 Iterative algorithm

We propose an iterative refinement for the spectral decomposition $H = Q\Lambda Q^T$ of a given real symmetric matrix H of order n. As we know, the more accurate Jacobi methods are several times slower than the standard methods, based on tridiagonalisation.

Our aim is, therefore, to solve the problem as quickly and as accurately as possible. However, the existing algorithms can not satisfy both of these two requirements, thus we have to find a compromise, which ultimately depends on our priorities.

To satisfy the speed requirement, Drmač and Veselić [3] suggested to first perform the eigenvalue decomposition $H = Q\Lambda Q^T$ by using some fast method (QR or D&C), then to compute the Rayleigh matrix $H' = Q^T H Q$ and finally to refine the solution by applying the Jacobi method to the Rayleigh matrix. They proved that this algorithm is more efficient (faster) then Jacobi method on the initial matrix H, while giving the similar relative accuracy as the Jacobi method.

In this paper we show that we get similar accuracy and faster convergence if in the third step we substitute Jacobi method with one of the fast methods and then iteratively repeat the second and the third step until accuracy requirement is satisfied. We suggest the following iterative algorithm:

Algorithm

1. decompose $H = Q\Lambda Q^T$ by a fast method,
2. compute the Rayleigh matrix $H' = Q^T H Q$,
3. if H' is not diagonal enough treat it again by a fast method $H' = Q'\Lambda' Q'^T$,
4. repeat 2. and 3. step for $Q := QQ'$.

In the third step of Algorithm (if H' is not diagonal enough) we use the following stopping criterion

$$E_{ij} \le \varepsilon \cdot tol \cdot \sqrt{|D_{ii}||D_{jj}|}, \quad \text{for all } 1 \le i, j \le n, \quad (4)$$

where $D = diag(H')$ and $diag(E) = 0$, if the matrix H' is written as $H' = D + E$. If condition (4) is not fulfilled for all elements of matrix H', we must find the partition

$$D_1 = D + E = \left[\begin{array}{c|c} D_{11} + E_{11} & E_{12} \\ \hline E_{12}^T & D_{22} + E_{22} \end{array}\right], \quad (5)$$

where elements of matrices E_{11} and E_{12} fulfil the condition (4) while elements of matrix E_{22} are relatively big in comparison with elements of matrix D_{22}. We can find the partition (5) in two different ways (for details see [5]):

1. by choosing "bad elements" (those elements which do not satisfy the condition (4)) or
2. by choosing "good elements" (those elements which satisfy the condition (4)).

Although the criterion by choosing good elements does not assure that all eigenvalues will be improved (some of them can even be spoiled), the numerical results show us that this strategy of good elements gives the eigensolution with high relative accuracy and it is more efficient than strategy of bad elements.

3 Numerical Experiments

In this section we present the results of our numerical experiments. We tested the Algorithm above for different classes of matrices. We generate three types of random matrices:

1. indefinite matrices,
2. positive definite matrices and
3. scaled diagonally dominant matrices.

For each dimension of matrix ($n = 6, 16, 32, 50, 100$) we generate 120 random matrices in the first two classes and 40 random matrices in the third class. This makes a total of 1400 different test matrices.

For a fast method in the first and the third step of Algorithm we use D&C method based on tridiagonalisation by Householder's reflectors (see [4]). We also tested QR method instead of D&C method and got similar accuracy. Because D&C method is quicker than QR we use it as a benchmark in further analysis.

In the third step of Algorithm we use the stopping criterion by choosing good elements. Although this criterion does not assure that all eigenvalues will be improved (some of them can even be spoiled), the numerical results from Table 1 show us that relative errors of computed eigenvalues on the average become smaller.

The reference values are computed by Jacobi method [8] and are denoted by λ_i. The computed eigenvalues are denoted by λ'_i and the relative error is computed as

$$relative\ error = \max_i \frac{|\lambda_i - \lambda'_i|}{|\lambda_i|}. \quad (6)$$

The computed relative errors are given in Table 1, Table 2 and Table 3. In Table 1 we use the following notations:

- MinHDC: the minimal relative errors for the first step of Algorithm,
- MaxHDC: the maximal relative errors for the first step of Algorithm,
- AveHDC: the average relative errors for the first step of Algorithm,
- AveALG: the average relative errors of Algorithm.

type	dim	MinHDC	MaxHDC	AveHDC	AveALG
1	*6*	5,0E-09	5,7E-07	1.01E-07	1.51E-07
	16	2,2E-13	2,3E-04	2.12E-05	8.50E-07
	32	5,8E-15	2,8E-03	1.83E-04	3.18E-06
	50	9,6E-15	8,7E-03	3.04E-04	8.76E-05
	100	2,3E-14	2,1E-02	4.25E-04	9.85E-08
2	*6*	7,3E-16	2,3E-06	3.54E-07	6.75E-08
	16	1,7E-15	2,8E-03	1.85E-04	8.54E-08
	32	4,1E-15	5,9E-03	3.53E-04	2.75E-05
	50	6,0E-15	2,1E-02	6.47E-04	8.41E-05
	100	1,5E-14	2,8E-02	6.63E-04	5.83E-05
3	*6*	2,7E-12	2,5E-05	4.30E-06	5.58E-07
	16	2,6E-13	4,2E-03	9.12E-07	8.00E-10
	32	3,7E-15	4,4E-03	2.80E-06	1.53E-08
	50	5,4E-15	1,6E-02	1.82E-06	3.66E-07
	100	1,3E-14	2,1E-02	2.45E-05	1.54E-06

Table 1

In Table 2 and Table 3 we use the following notations:

- NImp: the number of improved eigenvalues,
- MinImp: the minimal relative errors of improved eigenvalues,
- MaxImp: the maximal relative errors of improved eigenvalues,
- NSpo: the number of spoiled eigenvalues,
- MinSpo: the minimal relative errors of spoiled eigenvalues,
- MaxSpo: the maximal relative errors of spoiled eigenvalues.

type	*dim*	*NImp*	*MinImp*	*MaxImp*
1	*6*	1,7	1,7E-08	1,8E-08
	16	6,1	9,3E-12	1,7E-05
	32	11,1	1,6E-14	1,9E-05
	50	18,0	2,7E-14	1,8E-04
	100	31,5	2,6E-14	5,5E-06
2	*6*	1,7	8,3E-11	8,5E-11
	16	4,6	3,6E-13	6,3E-07
	32	10,4	8,5E-13	7,5E-04
	50	18,8	7,8E-15	9,0E-06
	100	34,9	1,6E-14	2,5E-03
3	*6*	0,9	2,3E-13	2,0E-09
	16	5,5	8,0E-12	9,4E-07
	32	8,7	6,4E-13	5,6E-04
	50	15,0	1,3E-14	6,7E-06
	100	27,7	5,6E-12	1,9E-03

Table 2

type	*dim*	*NSpo*	*MinSpo*	*MaxSpo*
1	*6*	1,3	8,8E-07	1,3E-06
	16	1,5	4,8E-05	4,8E-05
	32	1,5	9,1E-05	9,2E-05
	50	1,3	2,7E-03	2,7E-03
	100	1,3	9,7E-03	9,7E-03
2	*6*	1,5	1,6E-07	5,5E-07
	16	3,1	1,3E-05	1,3E-05
	32	3,3	6,6E-06	7,5E-06
	50	2,2	1,8E-03	6,0E-03
	100	0,9	1,2E-02	1,3E-02
3	*6*	0,3	2,3E-05	2,3E-05
	16	1,8	1,9E-05	2,0E-05
	32	2,6	2,1E-05	2,2E-05
	50	1,7	1,4E-03	4,5E-03
	100	0,7	9,3E-03	9,8E-03

Table 3

The timing results for Jacobi method [8] (notation TimeJAC) and for our Algorithm (notation TimeALG) are given in Table 4. The number of iterations is denoted with NumIt. For all quantities we give mean value on the respective class and dimension of test matrices.

type	dim	*TimeJAC*	*TimeALG*	*NumIt*
1	*6*	1,3	2,0	0,9
	16	13,4	11,5	1,0
	32	85,7	59,2	0,9
	50	292,6	183,0	0,9
	100	2099,9	1150,0	0,8
2	*6*	1,2	1,9	1,0
	16	13,2	12,2	1,2
	32	80,3	64,0	1,2
	50	269,6	199,4	1,3
	100	1880,4	1245,7	1,2
3	*6*	1,0	1,4	0,4
	16	8,2	9,8	1,1
	32	45,2	49,7	1,0
	50	145,7	154,6	1,1
	100	1053,5	1013,8	1,0

Table 4

As we can see from Table 2, we get about 30 % improved eigenvalues. Although we get about 8 % spoiled eigenvalues (Table 3), we can conclude that the maximal relative errors of spoiled values (the column MaxSpo of the Table 3) are still smaller than maximal relative errors of the initial calculated values (the column MaxHDC of the Table 1). This means that on the average relative errors become smaller as we can see in the last column (AveALG) of Table 1.

As we can see from Table 4, on the average only one iteration of Algorithm is needed for the refinement of the eigensolution. Moreover, we can conclude that our Algorithm is more efficient than Jacobi method. If we compare TimeALG and TimeJAC in Tabel 4, we can see that our Algorithm brought about 27 % speed up over Jacobi method.

4 Conclusion

Based on results of a series of numerical experiments we show that the combination of tridiagonalisation by using Householder's reflectors and D&C method, considering the stopping criterion by choosing good elements, gives us the most efficient (fastest) algorithm, taking into account the accuracy requirement.

References:

[1] J. Barlow and J. Demmel, Computing Accurate Eigensystems of Scaled Diagonally Dominant Matrices, *SIAM J. Numer. Anal.*, Vol.27, No.3, 1990, pp. 762-791.

[2] J. Demmel and K. Veselić, Jacobi's method is more accurate then QR, *SIAM J. Matrix Anal. Appl*., Vol.13, No.4, 1990, pp. 1204-1245.
[3] Z. Drmač and K. Veselić, Approximate eigenvectors as preconditioner*, Linear Algebra and its Appl. 309*, 2000, pp. 191-215.
[4] L. Ferbar, Computing the Eigenvalues by a Fast Method, *Proc. of the 6th International Symposium on Operational Research in Slovenia*, Preddvor, 2001, pp. 65-69.
[5] L. Ferbar, Stopping Criterion at the Iterative Refinement of the Eigensolution, *Proc. of Slovenian Informatics Conference 2002*, Portorož, 2002, pp. 350-354.
[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore and London : University Press, 1989.
[7] R. A. Rosanoff, J. F. Gloudeman and S. Levy, Numerical conditions of stiffness matrix formulations for frame structures, *Proc. of the 2nd Conference on Matrix methods in Structural Mechanics*, WPAFB, Dayton, 1968.
[8] I. Slapničar, *Accurate Symmetric Eigenreduction by a Jacobi Method*, PhD thesis, Lehrgebiet Mathematische Physik, Fernuniversität Hagen, 1992.
[9] K. Veselić and I. Slapničar, Floating-point perturbations of Hermitian matrices, *Linear Algebra and its Appl. 195*, 1993, pp. 81-116.
[10] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford : Clarendon Press, 1965.